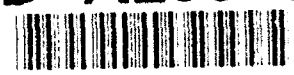
AD-A266 188



ATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	11/25/92	Final Report 10/1/91 - 9/30/92

4. TITLE AND SUBTITLE
ADAPTIVE ALGORITHM FOR AIRCRAFT CONFIGURATION IN TURBULENT FLOW

5. FUNDING NUMBERS
AFOSR-91-0032
2307/AS

6. AUTHOR(S)
Dr. John Kallinderis

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Dept. of Aerospace Engineering & Engineering Mechanics
The University of Texas at Austin
Austin, TX 78712-1085

8. PERFORMING ORGANIZATION REPORT NUMBER
CAR 92-8
AFOSR-TR-93 0433

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
AIR FORCE OFFICE OF SCIENTIFIC RESEARCH
DIRECTORATE OF AEROSPACE SCIENCES
BOLLING AFB, DC 20332-6448

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFOSR-91-0032

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE
DISTRIBUTION IS UNLIMITED

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

The flow around aircraft configurations has been simulated. A code for numerical simulation and study of 3-D aircraft flowfields has been developed. The code consists of a hybrid scheme which solves the Navier-Stokes equations within the viscous regions around the aircraft while the simpler Euler system is solved everywhere else. Efficient resolution of local flow physics such as shear layers, shock waves, and vortices is achieved by a specially developed adaptive grid algorithm. The method employs division and/or deletion of grid cells, as well as redistribution of points in order to resolve local flow features more accurately and efficiently. A hybrid grid consisting of prisms and tetrahedra grid-cells is employed. The first type is appropriate for the viscous regions, while the second type is suitable for the inviscid regions. A grid generator of prisms around an F-16-type aircraft was developed. Flow visualization is greatly enhanced with a developed computer graphics code.

93-14664

14. SUBJECT TERMS
Navier-Stokes numerical scheme, Euler numerical scheme, adaptive grid, F-16 aircraft grid generation, computer graphics, aircraft flow simulations.

15. NUMBER OF PAGES
56

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	SAR

Adaptive Algorithm for Aircraft Configurations in Turbulent Flow

Final Technical Report

to

Air Force Office of Scientific Research

Program Manager : Dr. L. Sakell

by

J. Kallinderis

Assistant Professor

Department of Aerospace Engineering and Engineering Mechanics

The University of Texas at Austin

November 1992

Contents

1 Research Project 1

1.1 Tasks 1

2 Results 3

3 Publications Acknowledging AFOSR 6

4 Technical Description of Accomplishments 8

1 Research Project

The present research project studied three-dimensional flow around aircraft configurations. The project has developed an algorithm that adapts the grid, the flow equations, and the numerical scheme to the evolving flow field. A general code (CODE3D) has been completed. CODE3D applies the turbulent Navier-Stokes equations within the viscous regions, while the simpler Euler equations are employed over the inviscid areas of the domain. The code generates grids with two different topologies. The viscous region is covered with prismatic cells, while the inviscid part is covered with tetrahedra. The method attains increased local grid resolution by subdividing cells and/or by redistributing nodes. The algorithm focuses on the complicated physics of aircraft flows. The emphasis is on local flow features, which include shear layers, shock waves, and vortices. The main applications were an F 16-type aircraft configuration, as well as a transport aircraft for which experimental data were available.

1.1 Tasks

The project was broken into eight tasks, which were pursued. Those tasks were as follows:

1. Development and validation of an Euler solver with tetrahedra grids.

2. Development and validation of a Navier-Stokes solver with prismatic grids.

3. Development of a method for adaptive grid refinement/coarsening of tetrahedra.

4. Development of a method for adaptive redistribution of semi-unstructured prismatic grids.

5. Development of grid generator for aircraft geometries.

6. Development of a flow visualization package with 3-D, unstructured grids.

7. Development of general interface between any type of aircraft geometry and the solver.

8. Aircraft flow simulations.

2 Results

The tool for the aircraft flow simulations has been a unified code that was developed as part of the project. The code consists of a hybrid solver (Navier-Stokes / Euler), generating a hybrid grid (Prisms / Tetrahedra), using two types of grid adaptation (Redistribution / Embedding). Specifically, the results of the project were:

1. *Euler code with tetrahedra grids*

   An Euler, Finite-Volume code (EU3D) for unstructured tetrahedral grids was developed. Accuracy, computing time, and storage requirement were evaluated via test cases, which included wing, and aircraft flows. Storage requirements were kept to a minimum.

2. *Navier-Stokes code with prismatic grids*

   A full Navier-Stokes, Finite-Volume code (NS3D) was developed for the semi-unstructured, prismatic grids that are employed to cover the regions close to the aircraft. The structure of the grid in one of the directions allowed drastic reduction in memory required by the code.

3. *Adaptive grid refinement/coarsening code for tetrahedra grids*

   A code (ADAP3D) for adaptive refinement / coarsening of tetrahedra grids was developed and evaluated. Anisotropic (directional) division of the cells allowed significant reduction in required number of cells. Applications of the method included adaptation of the tetrahedral initial grid around the low wing transport (LWT) aircraft provided by NASA.

4. *Adaptive redistribution of semi-unstructured prismatic grid*

   Redistribution of the nodes of the viscous regions prismatic grid was employed for better resolution of the viscous stresses. This was achieved by moving the nodes along the normal-to-surface direction.

5. *Prismatic grid generator around F-16-type of aircraft*

   An algebraic generator of prisms around aircraft configurations was developed. The computing time was one or two orders of magnitude lower than existing aircraft grid generators. A type of an F-16 aircraft was the main application.

6. *Tetrahedral grid generator around F-16-type of aircraft*

   An octree-type generator of tetrahedra around aircraft configurations was developed. The computing time was one or two orders of magnitude lower than existing tetrahedral grid generators. A type of an F-16 aircraft was the main application.

7. *Pre-processor*

   The code PRE3D was developed, which interfaces any aircraft geometry with the solver, and adaptive algorithm. It was designed to handle different grid topologies, such as prisms, and tetrahedra. The geometries for which it was applied included a version of the F-16 aircraft, and a low wing transport plane.

8. *Flow visualization with 3-D, unstructured grid graphics*

   A general package (PLOT3D) was developed for graphical representation of three-dimensional geometries, unstructured grids, as

well as of flow fields using Sun and IBM workstations. Among the features of the graphics code are cuts (planar and non-planar) through the grid and flow field, as well as contour plots on selected boundary and interior surfaces. Evolution of unsteady flow fields, as well as of the adaptive grid can also be visualized.

9. *Aircraft simulations.*

   Flow around two types of aircraft configurations was simulated. The first was an F-16 type, while the second was a low wing transport configuration. Experimental data from NASA was used in the latter case for code validation.

Impact of Research on CFD

The research of the project led to the following novel contributions to CFD:

1. Generation and use of prismatic grids for viscous flow simulations.
2. Generation of grid around aircraft that takes computing time of the order of a few minutes using a workstation.
3. One of the first adaptive grid simulation of viscous flow around aircraft.

3 Publications Acknowledging AFOSR

The related publications include:

1. Two MS theses.
2. Six refereed conference papers.
3. Four submitted refereed journal papers.

Specifically, the publications related to the project are:

1. Y. Kallinderis, V. Parthasarathy, and J. Wu, *A New Euler Scheme and Adaptive Refinement / Coarsening Algorithm for Tetrahedra Grids*, AIAA Paper 92-0446, Reno NV, January 1992.
2. Y. Kallinderis, *A New Finite-Volume Navier-Stokes Scheme on 3-D Semi-Unstructured Prismatic Elements*, AIAA Paper 92-2697, Palo Alto CA, June 1992.
3. Y. Kallinderis and S. Ward, *Prismatic Grid Generation with an Efficient Algebraic Method for Aircraft Configurations*, AIAA Paper 92-2721, Palo Alto CA, June 1992.
4. Y. Kallinderis and S. Ward, *Semi-Unstructured Prismatic Grid Generation for Aircraft Configurations*, Proceedings of the 13th Int. Conf. Num. Methods in Fluid Dynamics, July 1992.
5. P. Vijayan and Y. Kallinderis, *A 3-D Finite-Volume Scheme for the Euler Equations on Adaptive Tetrahedral Grids*, Journal of Computational Physics (in review).

6. P. Vijayan and Y. Kallinderis, *An Adaptive Refinement / Coarsening Scheme for 3-D Unstructured Meshes*, AIAA Journal (in review).

7. Y. Kallinderis and S. Ward, *Semi-Unstructured Prismatic Grid Generation for 3-D Complex Geometries with a Coupled Algebraic / Elliptic Method*, AIAA Journal (in review).

8. S. Ward and Y. Kallinderis, *Hybrid Prismatic / Tetrahedral Grid Generation for Complex 3-D Geometries*, AIAA Paper 93-0669, Reno NV, January 1993.

9. P. Vijayan and Y. Kallinderis, *A 3-D Finite-Volume Scheme for the Euler Equations on Adaptive Tetrahedral Grids*, AIAA 11th CFD Conference (in review).

10. S. Ward and Y. Kallinderis, *Tetrahedral Grid Generation for Complex 3-D Geometries*, AIAA Journal (in preparation).

11. P. Vijayan, *A Dynamic Grid Adaptation Scheme for 3-D Unstructured Grids*, MS Thesis, Dept. Aerospace Engineering, The University of Texas at Austin, Report No. CAR 92-6, May 1992.

12. S. Ward, *Hybrid Prismatic / Tetrahedral Grid Generation for Complex 3-D Geometries*, MS Thesis, Dept. Aerospace Engineering, The University of Texas at Austin, Report No. CAR 92-7, August 1992.

4 Technical Description of Accomplishments

Details on the algorithms developed, as well as on the simulations carried out are given in the following sections.

Hybrid Grid for the F-16 Aircraft

GENERATION OF PRISMATIC GRID

An unstructured triangular grid is employed as the starting surface to generate a prismatic grid. This grid, covering the body surface, is marched away from the body in distinct steps, resulting in the generation of structured prismatic layers in the marching direction . The process can be visualized as a gradual inflation of the body's volume. There are two main stages in the algebraic grid generation process. In the first, the destination of the marching surface is determined; in the second, the nodes are positioned on that surface. The marching surface is determined by employing a new technique based on *voxels* . The nodes are positioned by determining the *marching* vectors corresponding to the nodes of the previous surface.

Direct control of grid orthogonality and spacing is a main advantage of the algebraic method since the accuracy of Navier-Stokes solvers is critically dependent on these properties . Algebraic grid generators may yield a grid that is non-smooth and that may overlap. A grid generator must produce a grid with no overlapping faces, which may occur especially in concave regions. In both stages of the present algebraic method elliptic-type steps are employed in the form of Laplacian smoothing.

The goal of the marching scheme is to reduce the curvature of the previous marching surface at each step while ensuring smooth grid spacing

to avoid surface overlap. The developed method reduces surface curvature by marching the nodes to a known, smooth surface which encloses the current marching surface. The marching surface is modeled by its *voxel* representation, which is then smoothed.

2.1 Smooth Voxel Generation

A voxel is a three-dimensional element used to approximate a point in space. It may have any shape which will entirely fill a domain when voxels are placed adjacent to each other, thus conserving the volume. In the present work, parallelepiped voxels are used. The voxel representation of an object comprises all voxels partially occupying any part of the object. Figure 2 shows the voxel representation of the surface of an F-16A aircraft.

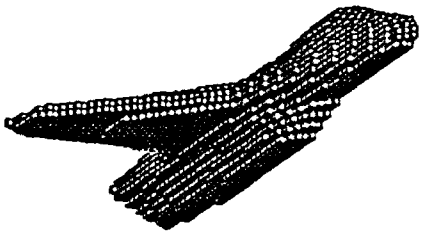


Figure 2: Voxel Representation of F-16A

In general, the voxel representation is externally bounded by quadri-

lateral faces which do not intersect the object. It is this aspect of voxel representations which allows them to be used to generate a conformal surface. Figure 3 shows the corresponding smoothed voxel representation of the same aircraft surface as in Fig. 2.

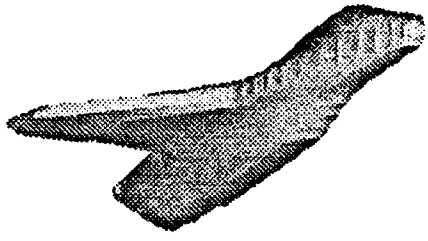


Figure 3: Smoothed Voxel Representation of F-16A

Three important facets of this method should be mentioned:

- The smoothing of the voxel face structure must be done in a manner which prevents voxel nodes from moving toward the marching surface. A voxel node is allowed to move only if its new position is visible from the neighboring faces. Laplacian-type of smoothing is applied a number of times (typically 10-20). The resulting smooth surface will be termed the *target* surface. Nodes forming the triangular faces of the prismatic grid need to be placed on this surface.

- The dimensions of the voxel should be such that surface details are captured. Excessively large voxels will "ignore" small scale surface features.

- If a large number of smoothing operations is performed, the voxel nodes move far from their original positions. This has the same effect as choosing a voxel dimension too large to capture surface detail.

It should be emphasized that the resulting *target* surface has reduced surface curvature compared to the marching surface. Concave regions are "filled-in", while convex areas are smoother.

2.2 Node Normals and Marching Vectors

One main issue of the present method is determination of the vectors along which each node of the triangular surface of the previous layer of prisms will march (*marching* vectors). The initital step is calculation of the vectors that are normal to the surface. For a continuous analytic surface, the calculation of the surface normals is trivial. However, the grid marching surface comprises discrete faces of discontinuous slope. Furthermore, nodes that lie on concave surfaces must march away from the concavity. Also, as the node marches, it must not intersect the current marching surface prior to intersecting the target surface.

This situation is avoided by enforcing a *visibility condition* which constrains the marching vectors such that the new node position is visible from all faces of the manifold. This region is the *visibility region*. It has the shape of a polyhedral cone extending outward from the node as shown in (Figure 4).

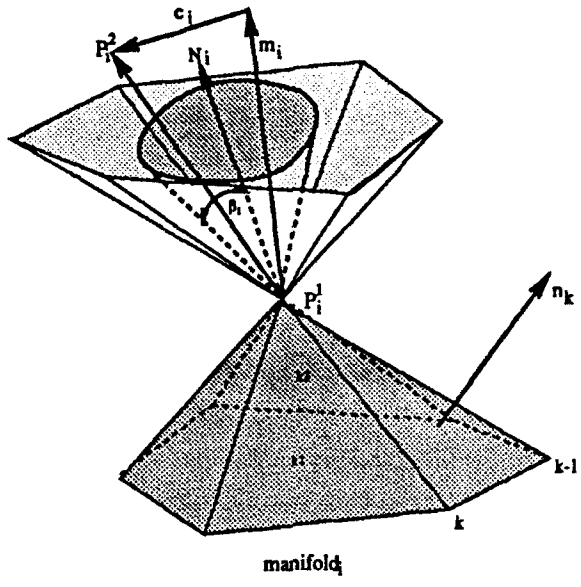


Figure 4: Visibility region

To simplify the constraints, a *visibility cone* with a circular cross-section and half-cone angle β centered on the normal vector is constructed at the node. This cone lies completely within the visibility region (see Figure 4).

The initial marching vectors $\hat{m}_i$ are the normal vectors $\hat{N}_i$. However, this may not provide a valid grid since overlapping may occur–especially in regions of the grid with closely spaced nodes. To prevent overlapping, the marching vectors must be altered to increase the divergence of all $\hat{m}_k$ on each manifold i. Therefore, a number of smoothing passes (typically 5) are performed over all the nodes on the marching surface:

$$\hat{m}_i = \frac{\sum_k \hat{m}_k}{K}. \tag{1}$$

After each step, a check is performed to determine if $\hat{m}_i$ lies outside the

visibility cone. The following equation is used:

$$(\hat{m}_i \cdot \hat{N}_i) \leq \cos(\beta_i). \tag{2}$$

If $\hat{m}_i$ does lie outside the visibility cone, then it is projected onto the cone and held fixed for subsequent operations.

2.3 Advancement and Smoothing of Marching Surface

The nodes on the previous surface are connected to the points of intersection of the *target* surface and the marching vectors $\hat{m}_i$. A special situation arises if a valid intersection cannot be found. The marching vector is then allowed to deviate within the visibility cone until a valid intersection with the target surface is found. If this fails, the degree of orthogonality constraint is successively relaxed and a search is made over the target faces to find the closest point to node i, while maintaining the visibility condition. If a valid intersection point still cannot be found, the target surface is not sufficiently conformal and the voxel dimension is reduced.

Marching to a point on the target surface may result in a reduction in face area caused by "convergence" of the marching grid lines. This will yield a non-uniform mesh; eventually, overlapping may occur. This situation may arise during growth of concave regions due to decrease in the surface area available for node placement. The nodes on the target surface are redistributed by applying a Laplacian-type operator so that the surface elements change their areas and curvature smoothly.

2.4 Distribution along the Marching Lines

Flexibility in specifying grid-spacing along the marching lines is crucial for accuracy of Navier-Stokes computations. The marching process generates a relatively small number of prisms occupying a relatively large volume and is therefore an undesirable grid for Navier-Stokes computations. This is remedied by generating new prisms with user-specified dimensions *within* the original prisms.

The prisms generated by marching the surface form a skeleton mesh of valid prisms within which any number of new cells may be generated. A scheme is employed which distributes new nodes along each one of the marching lines emanating from points on the body surface. In other words, the marching directions are maintained, but the marching distances (Δn) are modified. This is accomplished by performing a cubic-spline fit to each of the marching lines using the prism node locations for the spline knots. New nodes are then distributed along the splined lines. The distribution is such that the new node positions satisfy certain grid spacing requirements. In the present work, the spacing of the first point off the body surface is specified along with a constant stretching factor ω. Then, the n-positions of the points are given by the formula:

$$n_{j+1} = n_j + \omega(n_j - n_{j-1}). \tag{3}$$

A typical value of ω is 1.1.

3 Generation of Tetrahedral Grid

The tetrahedral grid forming the outer region of the computational domain is created through successive octree refinement. In this process,

an initial hexahedron is subdivided into eight smaller hexahedra called octants. Any octant which contains a portion of the *octree seed* is a *boundary octant* and is subdivided further (inward refinement). The resulting structure provides an efficient search path for the determination of new boundary octants. The hexahedral grid is further refined in a *balancing* process to prevent neighboring octants whose depth differs by more than one (outward refinement). The boundary octants are then altered to provide a smooth transition across the junction. Finally, all hexahedra are divided into tetrahedra to produce the outer grid. The resulting tetrahedral grid varies smoothly from the outermost prismatic layer to the domain outer boundary. Simplicity, grid quality, and the ability to match a pre-specified surface triangulation are the main advantages of the tetrahedral grid generation method presented here.

3.1 Octree Refinement and Data Structure

Efficiency of octree refinement relies heavily on the data structure defining the octree. For this reason, both global and local pointer structures were developed (see Figs 5 and 6). Each global octant comprises node, face and edge components.

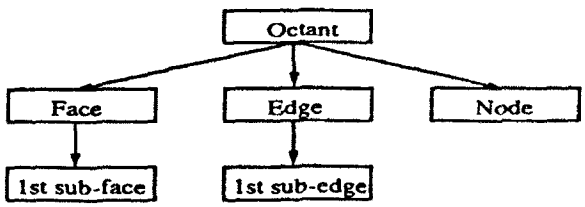


Figure 5: Global Octant Structure

The global pointer system used requires 26 words per octant, 5 words per face, and 3 words per edge. The local octant comprises sub-octant,

node, sub-face, and sub-edge components.

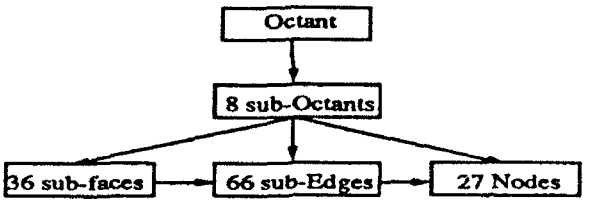


Figure 6: Local Octant Structure

The local octant structure requires a *total* of 129 words and provides the template for octant refinement. The same local octant is used for all global octant refinement. If an octant is to be subdivided, all external subcomponents of the local structure must first be defined. Any components of the global octant which have yet to be refined are subdivided. Then the values of the *local* subcomponents are obtained directly from the *global* pointers. This direct referencing completely avoids global searches for required data. After completion of the local structure, the sub-octants are constructed.

3.2 Inward Refinement

The criterion for octant subdivision is based on octant dimension and boundary containment. The refinement process is terminated once a characteristic dimension of the boundary octants has reached a specified threshold value. In general, this value should be such that a boundary octant will not contain whole triangular faces at the junction. Typically, this value is half the average prismatic grid edge length. The second part of the refinement criterion is boundary containment. Only boundary octants are subject to inward subdivision. Since a typical junction may comprise several thousand triangular faces, checking each octant for

boundary containment may be computationally expensive. However, a simple and efficient method to check for boundary containment has been developed.

The identification of boundary octants is reduced from a problem of multiple real-number calculations to determine octant containment of triangles in 3-D space at each refinement step to one of simple integer comparisons. This is accomplished by using the voxel representation of the junction layer. The voxel representation may be stored in terms of integer triplets defining the location of each voxel. Since this voxel is invariant throughout the generation process it needs to be constructed only once. Each octant is assigned an integer triplet domain (the equivalent integer range required to fill the octant with voxels) based on the octant dimension. If any voxel triplet is contained within the octant triplet domain, the octant is a boundary octant. Additionally, the structure of the octree allows associations between voxels and octants which limit the extent of the searches performed. This is a crucial factor for an efficient method.

3.3 Outward Refinement

It is desireable to have smoothly varying cells within the grid domain. Outward refinement is performed for two reasons: 1) to ensure that the resulting grid varies smoothly from the body surface to the outer domain boundary and 2) to ease the division of interface octants into tetrahedra. Figures 7 and 8 illustrate the effect of outward refinement. In Figure 7, the hexahedra change size rapidly while in Figure 8, the change is much more gradual.

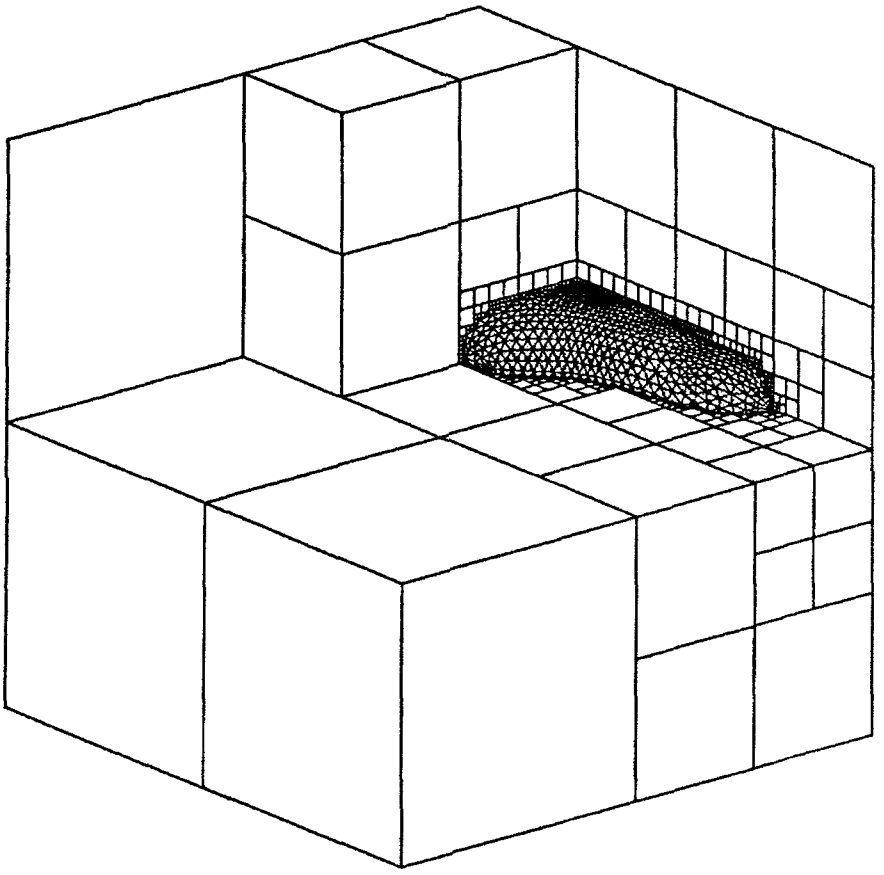


Figure 7: 3-D Section of domain showing seed surface and hexahedra without outward refinement

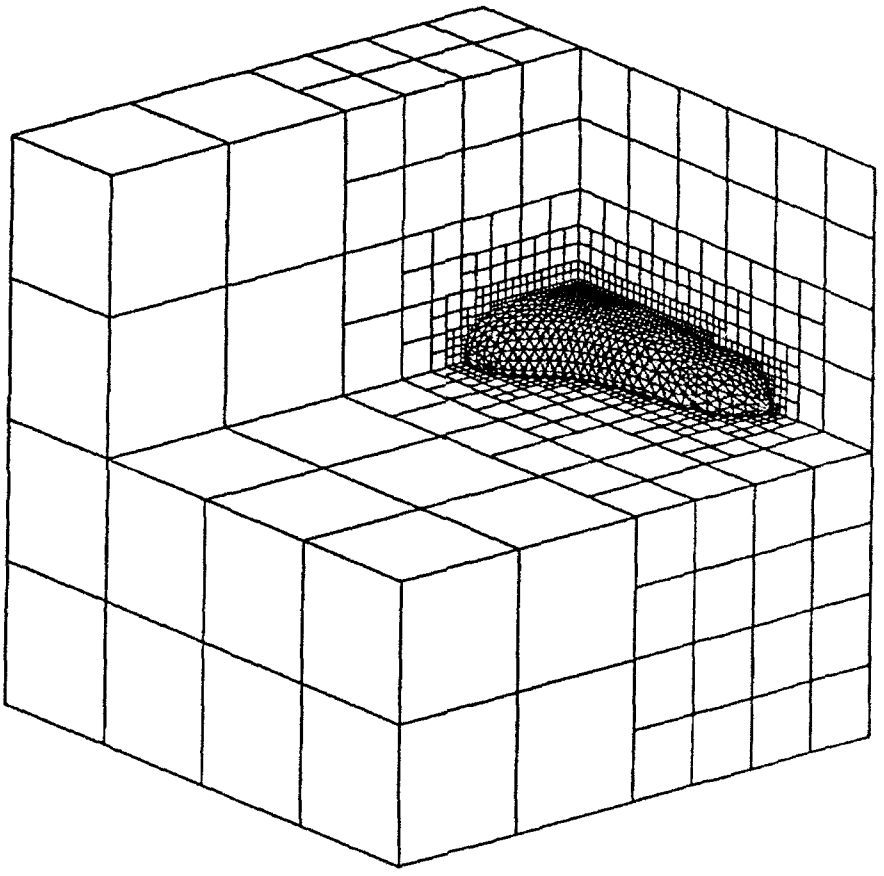


Figure 8: 3-D Section of domain showing seed surface and hexahedra with outward refinement

Figures 9 and 10 show the configuration of an interface octant before and after outward refinement, respectively. In Figure 9, one of the octant faces has been inwardly subdivided several times. It is seen that an octant may have an arbitrary number of subfaces in an arbitrary configuration on a given face after inward refinement. This presents difficulties in subsequent tetrahedral division as all nodes would have to be triangulated. The resulting tetrahedra would be ill-formed in most cases. In Figure 10, the original octant has been subjected to outward refinement. During the process, new octants are generated within interface octants until less than four subfaces occur on any face of the interface octant.

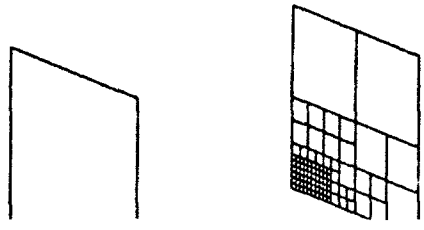


Figure 9: 2 faces of an interface octant showing face subdivision after inward refinement

The sole criteria for outward refinement is a difference greater than one between the depth of *any* octant component and the octant itself. Outward refinement is done in an iterative manner to allow for any propagation which may occur due to the refinement itself. Sweeps are made through the domain until no octants meet the refinement criterion. Octants which have any component at a depth greater than the octant are termed *interface* octants. Examples of interface octants are shown in Figure 11.

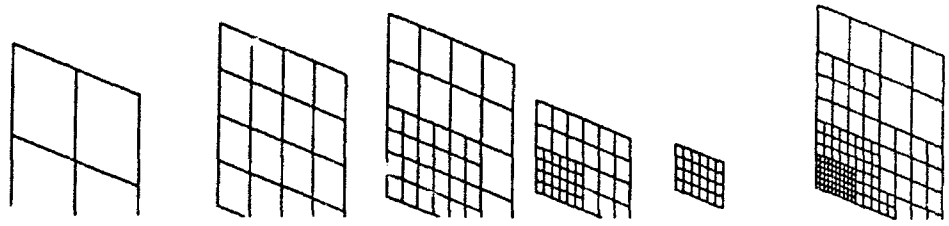


Figure 10: Outward refinement has reduced the complexity of hte interface octant by limiting the number of sub-faces on any face of the interface octant to 4

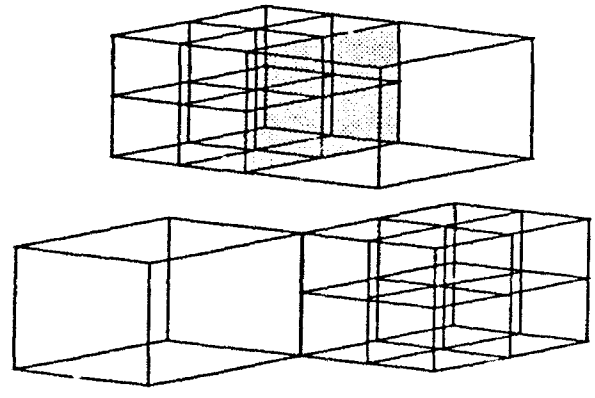


Figure 11: Face and Edge Depth Differences within an Octant

3.4 Tetrahedral Octant Refinement

Following division of the domain into smoothly varying octants, each octant must be subdivided into tetrahedra. It is possible to divide a hexahedron into either five or six tetrahedra. Since the tetrahedra must be consistent across octant faces, six tetrahedra are formed within each octant. This ensures octant to octant compatibility in the simplest manner because the same template may be used for all octants. (See Figure 12)

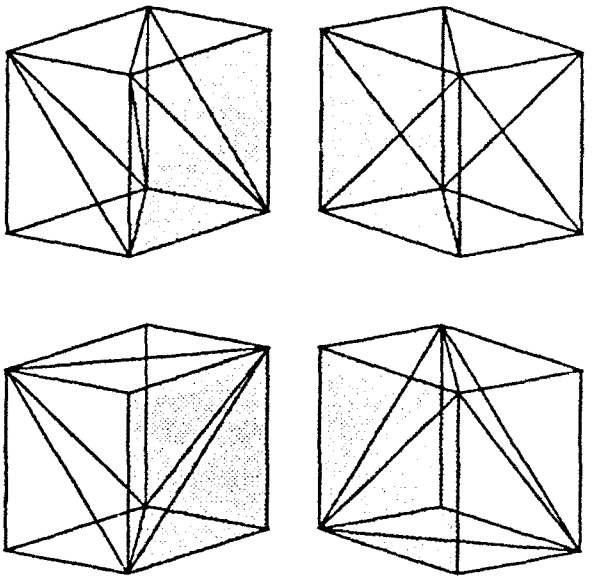


Figure 12: Division of hexahedra into tetrahedra. The upper octants have been divided into 6 tetrahedra and are consistent across the shaded afce. The lower octants have been divided into 5 tetrahedra and are inconsistent across the shaded face

Likewise, interface octant subfaces are divided according to a local template to ensure consistency with tetrahedra formed by non-interface octants.

Boundary octants are treated in a two-part process to reduce the complexity of the grid and at the junction. This is done by first moving boundary octant edges lying close to triangular surface edges to intersect the nearest triangular edge. Then, boundary octant nodes lying close to triangular faces are made to lie on the closest triangular face. Next, the faces of the boundary octant which intersect the junction are clipped at the intersection to form a boundary polyhedra. This has the effect of reducing the complexity of the junction and also helps to eliminate ill-formed cells. Finally, the boundary polyhedra is specially tesselated with tetrahedra in a way which eliminates the hanging nodes produced by the intersection. The final tetrahedral grid is shown in Figure 16 and comprises 168568 tetrahedra and 28302 nodes matching a seed surface of 2408 triangular faces.

4 APPLICATIONS

An F-16A aircraft geometry was chosen as a case for the developed grid generator, since the complexity and singularities of the surface are a severe test for the method. The main features of the configuration are the forebody, canopy, leading-edge strake, wing, shelf regions, and inlet. The surface triangulation consists of 2408 triangular faces , and of 1259 points. Generation of a prismatic grid around this geometry is quite complex, and especially so at the junctions between the different aircraft components. Two parts of the generated prismatic grid require examination in terms of quality. The first is the grid formed by the triangular faces of the prisms (unstructured part), while the second is the grid formed by the quadrilateral faces (structured part).

Two views of the initial and grown surfaces are shown in Figures 13 and 14. The growth of the grid is illustrated after 15 marching steps. The effect of the marching process is similar to inflating of the original body volume. It is observed that the distribution of points on the grown surface is quite smooth. The highly singular regions at the aircraft nose, the wing leading and trailing edges, the wing tip, the canopy, as well as the inlet, have been smoothed-out on the grown surface. Furthermore, the grid spacing on the grown triangular surface is smoother compared to the initial triangulated body surface. The singular concave regions at the junctions between the wing and the fuselage, as well as between the engine inlet and the body have been "filled-in", and the grid is more uniform over those regions compared to the initial grid on the body.

A view of the structured part of the prismatic grid is shown in Fig. 15. It should be noted that the depicted surface is not planar. The grid spacing on the quadrilateral surface is quite uniform. Furthermore, the marching lines emanating from the aircraft surface are quite smooth, including the ones that correspond to singular points. The lines do not have the "kinks" that appear frequently with algebraic grid generators.

A 3-D view of the outer tetrahedral grid is shown in Figure 16. It is seen that the tetrahedra vary quite smoothly in size from the outermost prism layer to the farfield.

The final number of prismatic cells is 96320 and the number of prismatic grid nodes is 50360. The final number of tetrahedra is 168568 and the number of tetrahedral grid nodes is 28302.

The required computing time for the prismatic grid was 1742 seconds while the time required for the tetrahedral grid was 88 seconds.

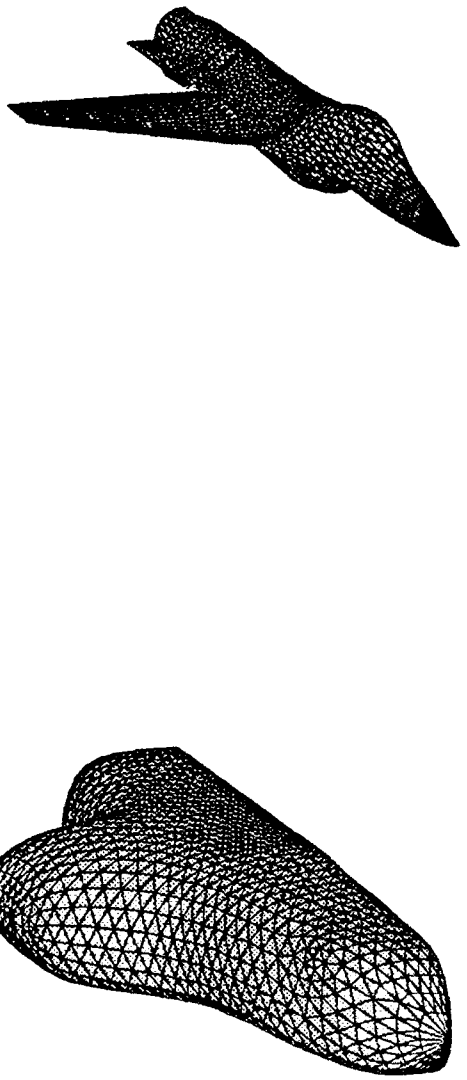
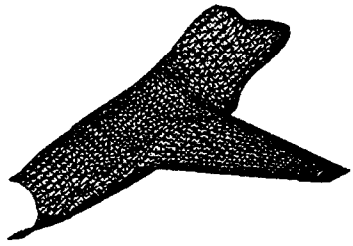


Figure 13: Foreview of Initial and Outermost Prism Surface (unstructured part of prismatic grid)



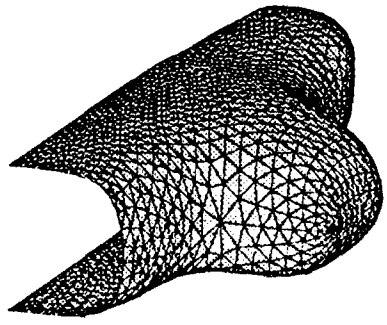


Figure 14: Aftview of Initial and Outermost Prism Surface (unstructured part of prismatic grid)

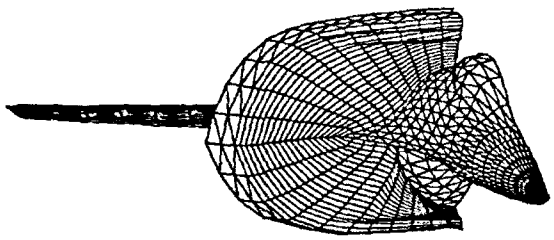


Figure 15: Structured part of the mesh at the junction between body and strake

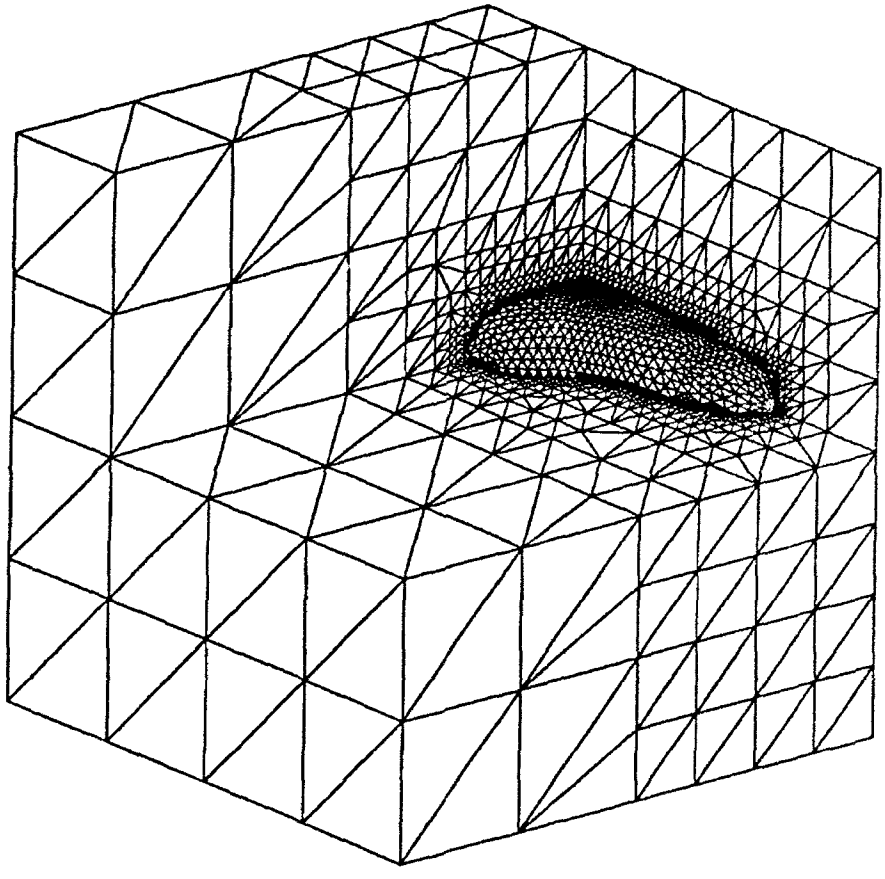


Figure 16: 3-D Section of domain showing seed surface and final tetrahedral grid

Navier-Stokes Method

2 NUMERICAL SCHEME

The Navier-Stokes equations of viscous flow are given in integral form for a bounded domain V as follows:

$$\frac{d}{dt}\iiint_V \mathbf{U}dV+\iiint_V \left(\frac{\partial(\mathbf{F}_I-\mathbf{F}_V)}{\partial x}+\frac{\partial(\mathbf{G}_I-\mathbf{G}_V)}{\partial y}+\frac{\partial(\mathbf{H}_I-\mathbf{H}_V)}{\partial z}\right)dV=0, \tag{1}$$

where

$$\mathbf{U}=\begin{pmatrix}\rho\\ \rho u\\ \rho v\\ \rho w\\ E\end{pmatrix},\mathbf{F}_I=\begin{pmatrix}\rho u\\ \rho u^2+p\\ \rho uv\\ \rho uw\\ (E+p)u\end{pmatrix},\mathbf{G}_I=\begin{pmatrix}\rho v\\ \rho vu\\ \rho v^2+p\\ \rho vw\\ (E+p)v\end{pmatrix}\mathbf{H}_I=\begin{pmatrix}\rho w\\ \rho wu\\ \rho wv\\ \rho w^2+p\\ (E+p)w\end{pmatrix}$$

are the state and convective flux vectors in the x, y, and z- directions respectively.

The viscous flux vectors are

$$\mathbf{F}_V = \begin{pmatrix} 0 \\ \tau_{xx} \\ \tau_{xy} \\ \tau_{xz} \\ u\tau_{xx} + v\tau_{xy} + w\tau_{xz} - q_x \end{pmatrix}, \mathbf{G}_V = \begin{pmatrix} 0 \\ \tau_{xy} \\ \tau_{yy} \\ \tau_{yz} \\ u\tau_{xy} + v\tau_{yy} + w\tau_{yz} - q_y \end{pmatrix} \tag{2}$$

, and

$$\mathbf{H}_V = \begin{pmatrix} 0 \\ \tau_{xz} \\ \tau_{yz} \\ \tau_{zz} \\ u\tau_{xz} + v\tau_{yz} + w\tau_{zz} - q_z \end{pmatrix},$$

where $\tau_{xx}, \tau_{yy}, \tau_{zz}, \tau_{xy}, \tau_{xz}, \tau_{yz}$ are the viscous stresses. For a perfect gas the pressure is related to the specific internal energy e by the relation $p = (\gamma - 1)\left[E - \frac{\rho}{2}(u^2 + v^2 + w^2)\right]$

After nondimensionalizing the above equations, the Mach and Reynolds numbers appear as parameters. Sutherland's law is used for computing the dynamic viscosity coefficient.

2.1 Finite-Volume Spatial Discretization

The number of nodes in a prismatic mesh is approximately half the number of cells. As a consequence, a vertex-based scheme appears to

require less storage compared to a cell-centered scheme. Minimization of storage requirement is one of the main issues in development of the present scheme, which stores the state-vector values at grid-nodes. Consider the typical grid-cell of Fig. 2, which has two triangular and three quadrilateral faces, where $\hat{i}, \hat{j}, \hat{k}$ are the unit vectors in the x,y,z-directions, and $\hat{n}$ is the unit vector normal to the cell-surface ∂V. The volume integral containing the spatial derivatives in equation (1) is equivalent to a surface integral via the divergence theorem:

$$\int\int\int_{\mathrm{V}} \left(\frac{\partial(\mathbf{F}_I - \mathbf{F}_V)}{\partial x} + \frac{\partial(\mathbf{G}_I - \mathbf{G}_V)}{\partial y} + \frac{\partial(\mathbf{H}_I - \mathbf{H}_V)}{\partial z} \right) dV =$$

$$\int\int_{\partial V} \left((\mathbf{F}_I - \mathbf{F}_V)\hat{i} + (\mathbf{G}_I - \mathbf{G}_V)\hat{j} + (\mathbf{H}_I - \mathbf{H}_V)\hat{k} \right) \hat{n} dS \qquad (3)$$

The above surface integral is discretized as:

$$\sum_{f=1}^{5} \left[(\mathbf{F}_I - \mathbf{F}_V)S_x + (\mathbf{G}_I - \mathbf{G}_V)S_y + (\mathbf{H}_I - \mathbf{H}_V)S_z \right]_f \qquad (4)$$

where the summation is over the five faces of the prism and S_x, S_y, S_z are the face-areas projected on the yz, xz , and $xy-$planes, respectively. The flux vectors are considered at the face-centers (f), and their values are obtained by averaging the values at the face-nodes (n). For each one of the two triangular faces, it is

$$(\mathbf{F}_I - \mathbf{F}_V)_f = \frac{1}{3} \sum_{n=1}^{n=3} (\mathbf{F}_I - \mathbf{F}_V)_n, \qquad (5)$$

while for each one of the quadrilateral faces the averaging is

$$(\mathbf{F}_I - \mathbf{F}_V)_f = \frac{1}{4} \sum_{n=1}^{n=4} (\mathbf{F}_I - \mathbf{F}_V)_n, \qquad (6)$$

where the summation is over the three nodes (n) of the face (f). Similar expressions are used for the $\mathbf{G}, \mathbf{H}$ terms.

2.2 Viscous Terms Evaluation Using Dual Cells

The viscous terms $\mathbf{F}_V, \mathbf{G}_V, \mathbf{H}_V$ consist of first order derivatives of the state-variables, which need be evaluated at the nodes. In the present work, evaluation of the viscous terms employs special cells that surround each node. Those cells will be referred to as *dual* cells The basic requirements on the *dual* mesh are:

1. each *dual* cell corresponds to only one grid-node.

2. each *dual* face is associated with one grid-face.

3. one only *dual* vertex lies within each grid-cell.

In order to construct the *dual* mesh each grid-cell is divided into six subcells of equal volume. Each triangular face of the prism is divided into three areas Node A at the center of the triangular face is connected with the middle of the edges dividing the face into three quadrilateral areas (Fig. 3). Also, each quadrilateral face of the prism is divided into two equal areas by connecting the middle of the edges that do not belong to the triangular faces (Fig. 3). Node C is the common node of all six subcells, which are hexahedra of equal volume. Each one of the six subcells constitutes a part of the *dual* cell that surrounds a particular node. For example, the hexahedron cell to which nodes 1 and C belong (Fig. 3) is a part of the *dual* cell that surrounds node 1. Each *dual* face shares exactly two neighboring cells. Figure 4 gives an example of a pair of a grid face with the corresponding *dual* face. The face is formed from nodes 1,3,4, while the *dual* face is formed by the center nodes C and C' of the two prisms that share face 1,3,4.

The viscous terms $\mathbf{F}_V, \mathbf{G}_V, \mathbf{H}_V$ consist of first order derivatives of flow variables, which are evaluated at the grid nodes. A typical derivative $\frac{\partial u}{\partial x}$ at node i is evaluated by considering a volume integral that is equivalent to a surface integral via the divergence theorem:

$$\frac{\partial u}{\partial x} = \frac{1}{V_i} \int\int\int_{V_i} \left(\frac{\partial u}{\partial x}\right) dV = \frac{1}{V_i} \int\int_{\partial V_i} u(\hat{i}.\hat{n}) dS, \tag{7}$$

with V_i being the volume of the *dual* cell corresponding to node i. The above surface integral is discretized as:

$$\sum_{f'} (uS_x)'_f \tag{8}$$

where the summation is over the faces of the dual cell which surrounds node i, and S_x is the face-area projected on the $yz-$plane. Figure 5 illustrates the two of the cells that are employed in order to update solution at node i, as well as the triangular faces of all prisms surrounding node i.

odd-even solution modes suppression

Odd-even modes frequently appear in numerical solutions due to non-linearities and/or shock-waves. Such solution modes are frequently transparent to the discrete equations, and they are acceptable steady state solutions. There are two different types of oscillatory modes that may be observed in solutions with prismatic elements. The first type are the modes on the unstructured (triangular) faces of the prisms, while the second type are the modes on the structured (quadrilateral) faces. Figure 6 shows the two possible oscillatory modes on triangular faces of the prismatic elements, where + and - indicate deviations from a mean. Sim-

ilarly, Fig. 7 illustrates the one possible such mode on the quadrilateral faces of the prisms.

The present choice of dual cells for evaluation of the viscous terms has the effect of suppressing the above odd-even modes. The value of the viscous terms contribution at node O of Figures (6), (7) will be negative, and it acts to cancel the positive odd-even mode value at that node.

2.3 Lax-Wendroff-Type Time Marching

The change in time of the state vector **U** at node i is evaluated through the following second order series expansion in integral form:

$$\iiint_{V_i} \delta \mathbf{U} dV \equiv \iiint_{V_i} (\mathbf{U}^{n+1} - \mathbf{U}^n) dV \cong$$

$$\iiint_{V_i} \Delta t_i (\frac{\partial \mathbf{U}}{\partial t})^n dV + \iiint_{V_i} \frac{(\Delta t_i)^2}{2} (\frac{\partial^2 \mathbf{U}}{\partial t^2})^n dV, \qquad (9)$$

which leads to the following equation:

$$\delta \mathbf{U}_i = (\frac{\Delta t}{V})_i \left(\iiint_{V_i} (\frac{\partial \mathbf{U}}{\partial t})^n dV + \iiint_{V_i} \frac{1}{2} \Delta t_i (\frac{\partial^2 \mathbf{U}}{\partial t^2})^n dV \right). \qquad (10)$$

The expansion is considered to be at the nodes and V_i denotes the dual cell volume that surrounds node i, while Δt_i is the corresponding time-step.

The time derivatives in equation 10 will be replaced by spatial derivatives according to the Lax-Wendroff approach:

$$\frac{\partial \mathbf{U}}{\partial t} = - \left(\frac{\partial (\mathbf{F}_I - \mathbf{F}_V)}{\partial x} + \frac{\partial (\mathbf{G}_I - \mathbf{G}_V)}{\partial y} + \frac{\partial (\mathbf{H}_I - \mathbf{H}_V)}{\partial z} \right), \qquad (11)$$

and

$$\frac{\partial^2 \mathbf{U}}{\partial t^2} = - \frac{\partial}{\partial t} \left(\frac{\partial (\mathbf{F}_I - \mathbf{F}_V)}{\partial x} + \frac{\partial (\mathbf{G}_I - \mathbf{G}_V)}{\partial y} + \frac{\partial (\mathbf{H}_I - \mathbf{H}_V)}{\partial z} \right) =$$

$$-\frac{\partial}{\partial x}\left(\frac{\partial(\mathbf{F}_I-\mathbf{F}_V)}{\partial \mathbf{U}}\frac{\partial \mathbf{U}}{\partial t}\right)-\frac{\partial}{\partial y}\left(\frac{\partial(\mathbf{G}_I-\mathbf{G}_V)}{\partial \mathbf{U}}\frac{\partial \mathbf{U}}{\partial t}\right)-\frac{\partial}{\partial z}\left(\frac{\partial(\mathbf{H}_I-\mathbf{H}_V)}{\partial \mathbf{U}}\frac{\partial \mathbf{U}}{\partial t}\right). \tag{12}$$

Evaluation of the viscous terms Jacobians $\frac{\partial \mathbf{F}_V}{\partial \mathbf{U}}$, $\frac{\partial \mathbf{G}_V}{\partial \mathbf{U}}$, and $\frac{\partial \mathbf{H}_V}{\partial \mathbf{U}}$, is quite expensive and complicated, especially when the extra terms due to a turbulence model are included . As a consequence, the viscous part of the second order in time term of eq. 9 is dropped, which results in a discretization that is first order accurate in time for the viscous terms, while second order temporal accuracy is kept for the inviscid terms. This results in a compact scheme, which has the same numerical molecule for both the inviscid, and the viscous terms. Moreover, all scheme operations are restricted to within each grid cell, which is crucial in cases in which adaptive grid refinement/coarsening is employed.

distributions to nodes with cell-based operations

It is important to arrange the scheme operations in such a way so that no information is required from outside of each cell. For this purpose, the numerical operations that are related to the *dual* mesh are split into cell-based operations.

The first order change-in-time of the state-vector at the cell-center $\Delta \mathbf{U}$ is defined as $\Delta t(\frac{\partial \mathbf{U}}{\partial t})^n$, and it is distributed to the six corners of the cells using equations 10, 11:

$$\delta \mathbf{U}_i = \frac{1}{8}(\frac{\Delta t}{V})_i(\frac{V}{\Delta t})_c \Delta \mathbf{U} \ , \ i = 1, ..., 6. \tag{13}$$

It is observed that the distributions are weighted by the factor $(\frac{\Delta t}{V})_i(\frac{V}{\Delta t})_c$. This term is close to one for uniform hexahedra grids. However, in the

case of unstructured grids the above term may differ significantly from unity, and it should be included in order to have a stable scheme.

The second order term of the Taylor series expansion 10 can be expressed using eqn 12, as follows:

$$\int\int\int_{V_i} \Delta t_i \left(\frac{\partial^2 \mathbf{U}}{\partial t^2}\right) dV = -\int\int_{\partial V_i} \Delta t_i \left(\frac{\partial \mathbf{F}_I}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial t}\right) \hat{i}\hat{n} dS$$

$$-\int\int_{\partial V_i} \Delta t_i \left(\frac{\partial \mathbf{G}_I}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial t}\right) \hat{j}\hat{n} dS - \int\int_{\partial V_i} \Delta t_i \left(\frac{\partial \mathbf{H}_I}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial t}\right) \hat{k}\hat{n} dS \qquad (14)$$

where V_i is the volume of the *dual* cell surrounding node i. Defining

$$\Delta \mathbf{F} \equiv \frac{\partial \mathbf{F}_I}{\partial \mathbf{U}} \Delta \mathbf{U} \ , \ \Delta \mathbf{G} \equiv \frac{\partial \mathbf{G}_I}{\partial \mathbf{U}} \Delta \mathbf{U} \ , \ \Delta \mathbf{H} \equiv \frac{\partial \mathbf{H}_I}{\partial \mathbf{U}} \Delta \mathbf{U}.$$

As stated, the viscous terms Jacobians have been omitted in the above expression for the second order in time term. Substituting into (14) it follows:

$$\int\int\int_{V_i} \Delta t_i \left(\frac{\partial^2 \mathbf{U}}{\partial t^2}\right) dV =$$

$$-\int\int_{\partial V_i} \Delta \mathbf{F}(\hat{i}\hat{n}) dS - \int\int_{\partial V_i} \Delta \mathbf{G}(\hat{j}\hat{n}) dS - \int\int_{\partial V_i} \Delta \mathbf{H}(\hat{k}\hat{n}) dS. \qquad (15)$$

$\Delta \mathbf{U}$ is evaluated at the center of the grid-cell, while $\Delta \mathbf{F}, \Delta \mathbf{G}, \Delta \mathbf{H}$ are evaluated at the grid-nodes. A correction term that accounts for the difference in time-step of the grid-cell (Δt_c) and of the *dual* cell (Δt_i) is employed in order to evaluate $\Delta \mathbf{F}, \Delta \mathbf{G}, \Delta \mathbf{H}$ from $\Delta \mathbf{U}$. Specifically, $\Delta \mathbf{U} = \frac{\partial \mathbf{U}}{\partial t} \Delta t_c$, while $\Delta \mathbf{F} = \frac{\partial \mathbf{F}_I}{\partial \mathbf{U}} \frac{\partial \mathbf{U}}{\partial t} \Delta t_i$. Therefore,

$$\Delta \mathbf{F} = \left(\frac{\Delta t_i}{\Delta t_c}\right) \frac{\partial \mathbf{F}_I}{\partial \mathbf{U}} \Delta \mathbf{U}. \qquad (16)$$

The same correction term is employed in a similar way for $\Delta \mathbf{G}, \Delta \mathbf{H}$. The

total distribution to node i then follows from equations (13),(15),(16) :

$$\delta \mathbf{U}_i = \frac{1}{8}(\frac{\Delta t}{V})_i(\frac{V}{\Delta t})_c \Delta \mathbf{U} - \frac{1}{8}(\frac{\Delta t}{V})_i \left[\sum_{fq=1, fq \neq i}^{3} (\Delta \mathbf{F} S_x + \Delta \mathbf{G} S_y + \Delta \mathbf{H} S_z)_{fq} \right] \tag{17}$$

$$- \frac{1}{6}(\frac{\Delta t}{V})_i \left[\sum_{ft=1, ft \neq i}^{2} (\Delta \mathbf{F} S_x + \Delta \mathbf{G} S_y + \Delta \mathbf{H} S_z)_{ft} \right].$$

In the above expression the first summation is over the quadrilateral faces (fq) of the prism, while the second summation corresponds to the triangular faces (ft) of the prism. Inclusion of the correction term is necessary for the stability of the scheme. Without it the scheme fails to converge.

The case considers supersonic flow of Mach number equal to 1.4, and of Reynolds number equal to 10^3 around a spherical body of maximum thickness equal to the radious. A prismatic mesh was generated from a hexahedra spherical mesh consisting of 20 nodes along the peripheral direction, 80 nodes in the azimuthal direction, and 30 nodes in the radial direction. The same boundary conditions were applied as in the previous case. Views of the mesh employed are illustrated in Figure 13. The symmetry plane (XY-plane), as well as the equatorial (XZ-plane) are shown. In addition Fig. 17 depicts a top view of the farfield boundary, which shows the unstructured part of the prismatic grid. Figure 18 shows the obtained solution in terms of Mach number contours. In this case, a spherical bow shock is observed upstream of the body.

Finally, figures 19 and 20 illustrate flow around an F-16 type of configuration. The Reynolds number is 10^6, the Mach number is 0.6, and the angle of attack is 0 degrees. A coarse grid consisting of 48K points was employed for the simulation. Figure 19 shows pressure contours on the upper surface, while Fig. 20 shows the paths of particles that were released at the nose region of the aircraft.

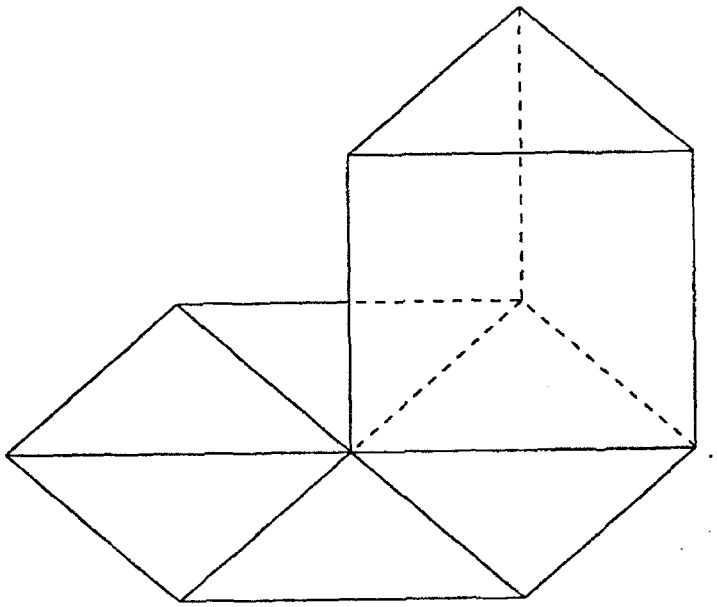


Figure 1: Semi-unstructured, prismatic grid

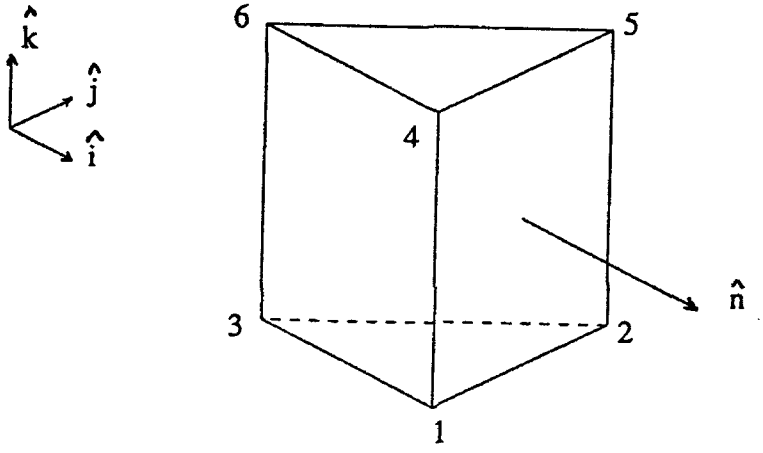


Figure 2: Prismatic grid cell notation

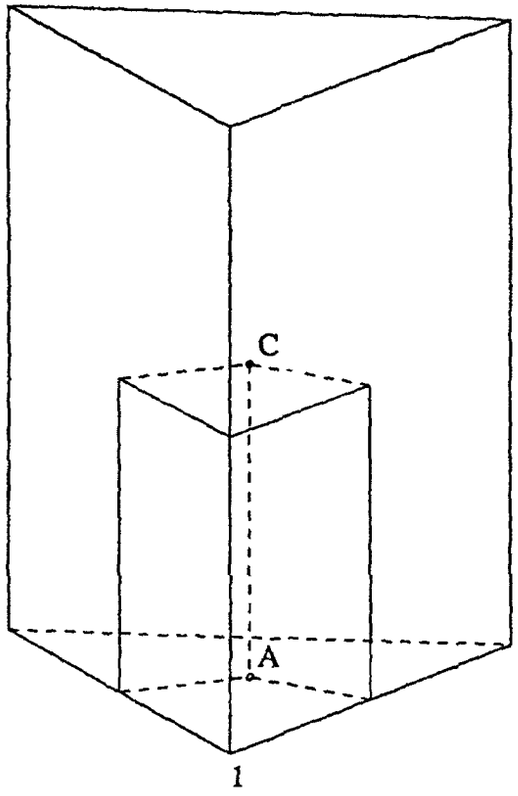


Figure 3: *Dual* cells construction by dividing the prism-cell into hexahedra subcells

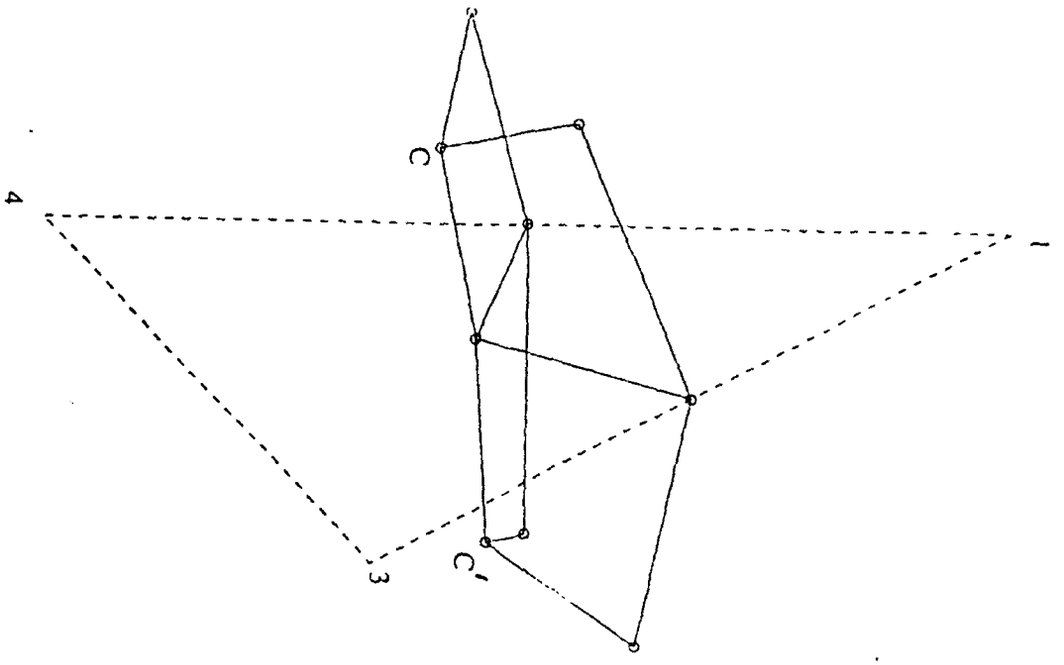


Figure 4: Pair of triangular face and corresponding *dual* face

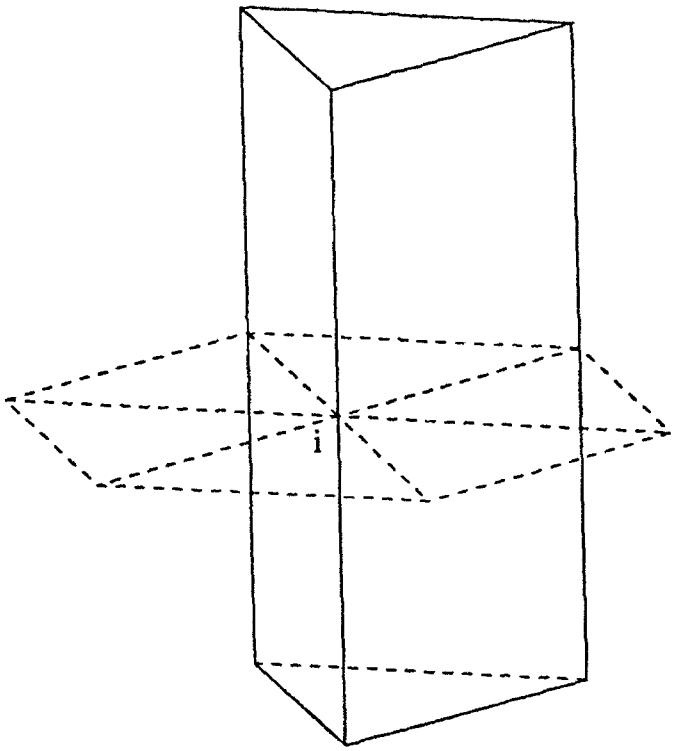


Figure 5: Control volumes corresponding to node i

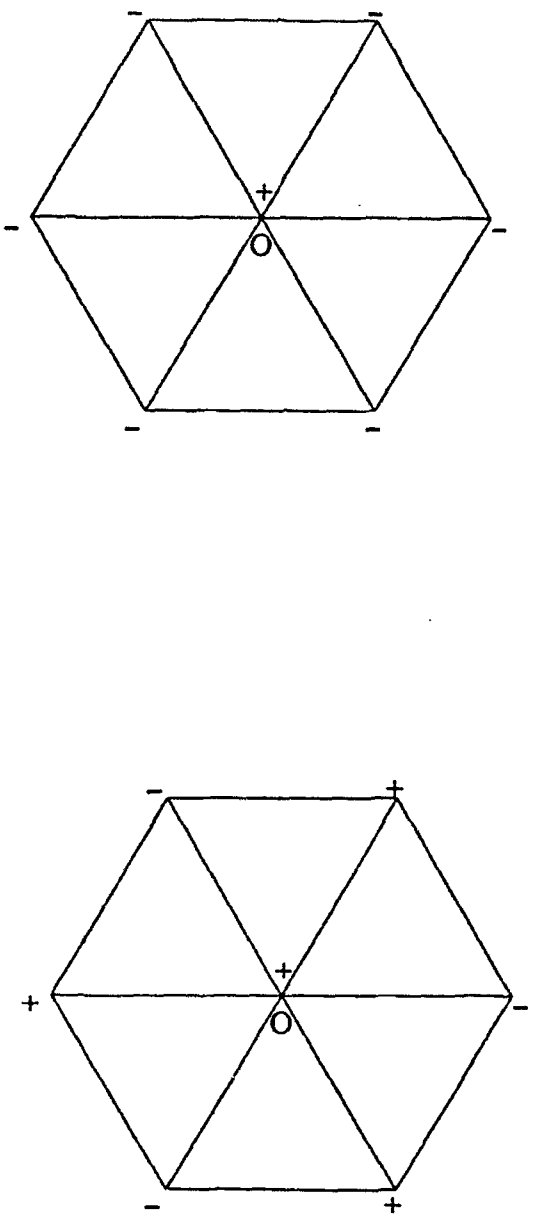


Figure 6: Possible odd-even modes on triangular faces of prisms

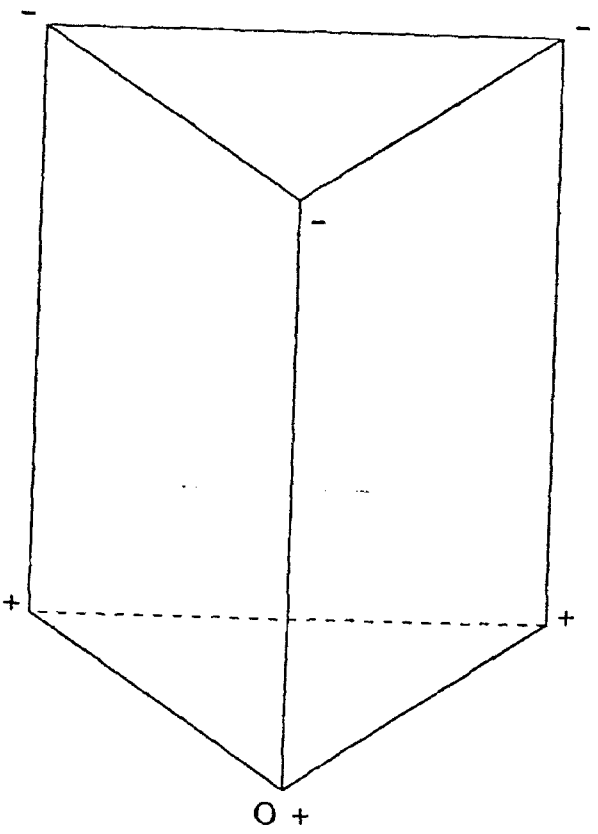


Figure 7: Possible odd-even modes on quadrilateral faces of prisms

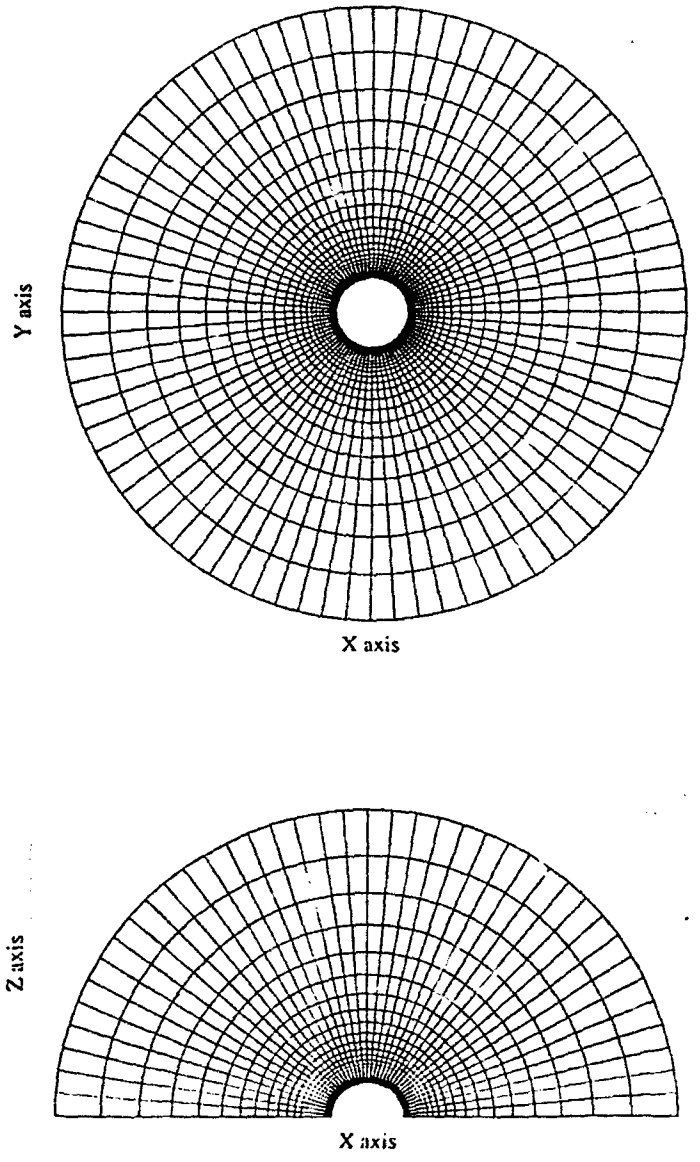


Figure 16: Two views of the spherical prismatic grid around large body created from a $80 \times 20 \times 30$ original hexahedra grid — Symmetry (XY) and equatorial (XZ) planes

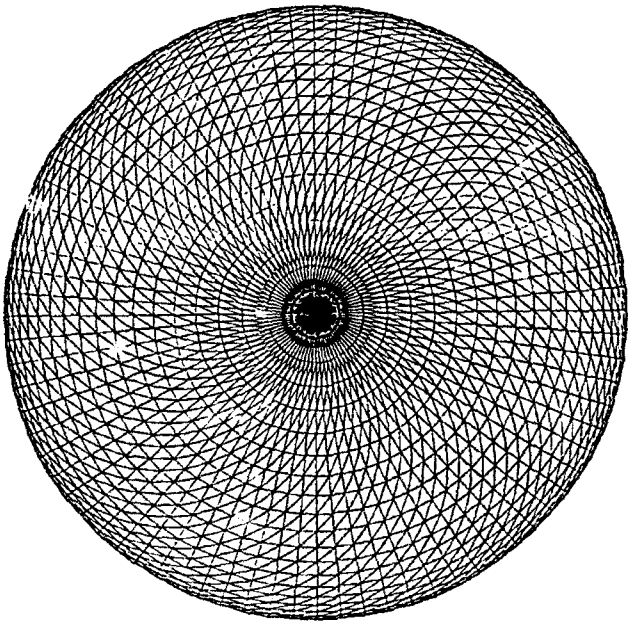


Figure 17: Top view of the farfield boundary showing the unstructured part of the prismatic grid

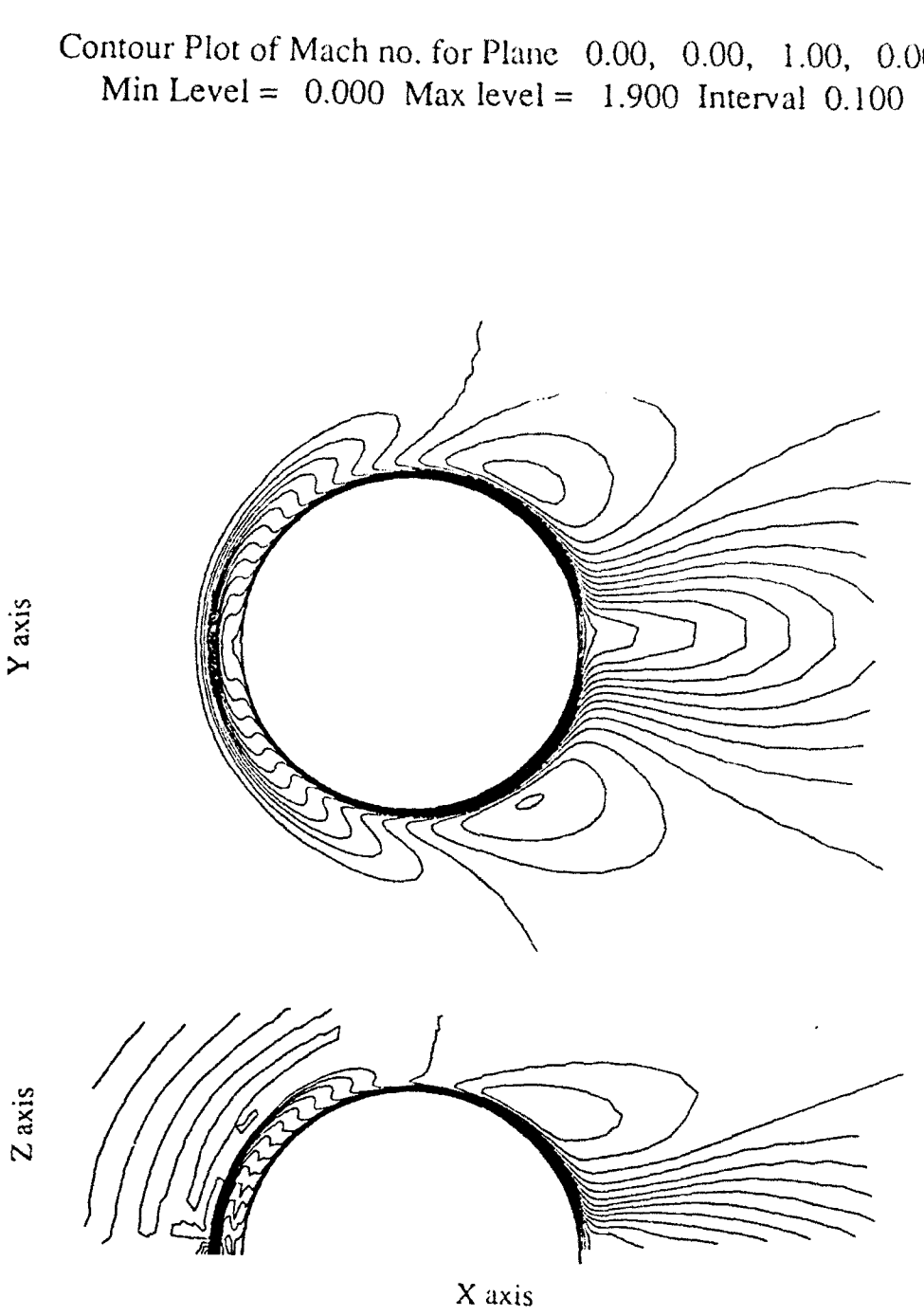


Figure 18: 3-D supersonic boundary layer and spherical bow shock around large body ($Re = 10^3, M_\infty = 1.4$) — Symmetry (XY) and equatorial (XZ) planes

Boundary Contour Plot of P Coeff

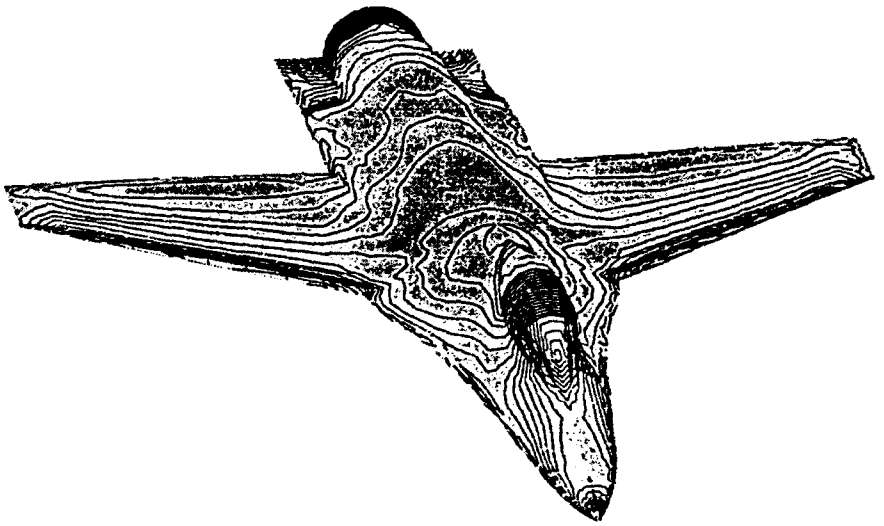


FIG. 19

Figure Paths of particles released at the nose region of the F16-type configuration

Adaptive Grid Method

2 Governing Equations

The governing equations to be solved is the system of time-dependent Euler equations for a perfect gas which combines the equations of mass, momentum and energy.

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} + \frac{\partial \mathbf{H}}{\partial z} = 0 \tag{1}$$

The state vector **U** and the flux vectors **F**,**G**,**H** are expressed in terms of the conservation variables, namely, density ρ, x,y,z- momentum and energy as follows:

$$\mathbf{U}^T = \begin{pmatrix} \rho & \rho u & \rho v & \rho w & E \end{pmatrix} \tag{2}$$

$$\mathbf{F} = \begin{pmatrix} \rho u \\ \rho u^2 + p \\ \rho uv \\ \rho uw \\ Eu + pu \end{pmatrix} \quad \mathbf{G} = \begin{pmatrix} \rho v \\ \rho vu \\ \rho v^2 + p \\ \rho vw \\ Ev + pv \end{pmatrix} \quad \mathbf{H} = \begin{pmatrix} \rho w \\ \rho wu \\ \rho wv \\ \rho w^2 + p \\ Ew + pw \end{pmatrix} \tag{3}$$

where E is the total energy, related to the other variables by an equation of state which, for a perfect gas, is

$$E = \frac{p}{\gamma - 1} + \frac{1}{2}\rho\left(u^2 + v^2 + w^2\right)$$

3 Temporal discretization

The solution at any particular node, say *0*, at time level *n+1* can be expressed in terms of the solution at time level n using a Taylor series expansion.

$$\mathbf{U}_0^{n+1} = \mathbf{U}_0^n + \delta\mathbf{U}_0^n$$

$$\delta\mathbf{U}_0^n = \mathbf{U}_0^{n+1} - \mathbf{U}_0^n = \left(\frac{\partial \mathbf{U}}{\partial t}\right)_0^n \Delta t + \left(\frac{\partial^2 \mathbf{U}}{\partial t^2}\right)_0^n \frac{\Delta t^2}{2} + \bigcirc\left(\Delta t^3\right) \qquad (4)$$

The temporal derivatives in the above expression are evaluated in terms of the spatial derivatives using the governing equations according to the Lax-Wendroff approach. The finite-volume method integrates the Euler equation (1) on the control volume Ω_0 enclosing a particular node, say *0*, which is enveloped by the boundary $\partial\Omega_0$.

$$\int_{\Omega_0} \left(\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} + \frac{\partial \mathbf{H}}{\partial z}\right) d\Omega = 0 \qquad (5)$$

which is rewritten using the divergence theorem as follows:

$$\int_{\Omega_0} \left(\frac{\partial \mathbf{U}}{\partial t}\right) d\Omega = -\int_{\partial\Omega_0} \left(\mathbf{F}n_x + \mathbf{G}n_y + \mathbf{H}n_z\right) dS \qquad (6)$$

$$\left(\frac{\partial \mathbf{U}}{\partial t}\right)_0 \Omega_0 = -\int_{\partial\Omega_0} \left(\mathbf{F}n_x + \mathbf{G}n_y + \mathbf{H}n_z\right) dS \qquad (7)$$

$$\left(\frac{\partial \mathbf{U}}{\partial t}\right)_0 = -\frac{1}{\Omega_0}\int_{\partial\Omega_0} \left(\mathbf{F}n_x + \mathbf{G}n_y + \mathbf{H}n_z\right) dS \qquad (8)$$

where n_x, n_y, n_z are the components of the unit vector normal to the area element dS of the boundary surface $\partial\Omega_0$.

The second order temporal derivative at the node *0* is evaluated as:

$$\int_{\Omega_0} \left(\frac{\partial^2 \mathbf{U}}{\partial t^2}\right) d\Omega = -\int_{\Omega_0} \frac{\partial}{\partial t}\left(\frac{\partial \mathbf{F}}{\partial x} + \frac{\partial \mathbf{G}}{\partial y} + \frac{\partial \mathbf{H}}{\partial z}\right) d\Omega \tag{9}$$

which, after changing the order of differentiation, is recast using the divergence theorem as follows:

$$\int_{\Omega_0} \left(\frac{\partial^2 \mathbf{U}}{\partial t^2}\right) d\Omega = -\int_{\partial\Omega_0} \left(\frac{\partial \mathbf{F}}{\partial \mathbf{U}}\frac{\partial \mathbf{U}}{\partial t}\right) n_x + \left(\frac{\partial \mathbf{G}}{\partial \mathbf{U}}\frac{\partial \mathbf{U}}{\partial t}\right) n_y + \left(\frac{\partial \mathbf{H}}{\partial \mathbf{U}}\frac{\partial \mathbf{U}}{\partial t}\right) n_z dS \tag{10}$$

$$\left(\frac{\partial^2 \mathbf{U}}{\partial t^2}\right)_0 = -\frac{1}{\Omega_0}\int_{\partial\Omega_0} \left(\tilde{\mathbf{A}} n_x + \tilde{\mathbf{B}} n_y + \tilde{\mathbf{C}} n_z\right) \frac{\partial \mathbf{U}}{\partial t} dS \tag{11}$$

where

$$\tilde{\mathbf{A}} = \left(\frac{\partial \mathbf{F}}{\partial \mathbf{U}}\right), \quad \tilde{\mathbf{B}} = \left(\frac{\partial \mathbf{G}}{\partial \mathbf{U}}\right), \quad \tilde{\mathbf{C}} = \left(\frac{\partial \mathbf{H}}{\partial \mathbf{U}}\right) \tag{12}$$

are the Jacobian matrices.

4 Spatial discretization

The surface integrals that appear in equations (8) and (11) are to be evaluated on the boundary of the control volume of each node in the grid. There are different approaches in defining the control volumes around the nodes.

4.1 Dual mesh

The spatial terms that appear in equations (8) and (11) are evaluated by employing special volumes that form the *dual* mesh. The dual

mesh is formed by constructing non-overlapping volumes, referred to as dual cells, around each node. The dual cells represent the control volume associated with the respective node. The dual mesh, for a two dimensional unstructured grid, is shown with dashed lines in Figure 1. The mesh is constructed by connecting the mid-points of the edges and the centroids of the triangular elements that constitute the grid and henceforth dividing each triangle into three quadrilaterals of equal areas. The finite-volume around any node, say *0*, is constituted by the union of all the quadrilaterals which share that node.

Analogously, the dual mesh for a tetrahedral grid is constructed by dividing each tetrahedron into four hexahedra of equal volumes, by connecting the mid-edge points, face-centroids and the centroid of the tetrahedron. Figure 2 shows a tetrahedron 0-1-2-3 with the two hexahedral cells 0-E1-F2-C-F3-E4-F1-E5 and 3-E2-F2-C-F4-E6-F1-E5 that constitute a portion of the dual cells around the nodes *0* and *3* respectively. The control volume around a node *0* is thus constituted by a polyhedral hull which is the union of all the hexahedra that share that node. The quadrilateral faces that constitute the dual mesh may not all be planar.

4.2 Flux evaluation using edge based operations

The surface integral term in Equation (8) represents the mass, momentum and energy flux across the faces of the control volume around the node *0*. The equation is written in the discrete form as follows:

$$-\frac{1}{\Omega_0}\int_{\partial\Omega_0}(\mathbf{F}n_x+\mathbf{G}n_y+\mathbf{H}n_z)\,dS=-\frac{1}{\Omega_0}\sum_k(\mathbf{F}S_x+\mathbf{G}S_y+\mathbf{H}S_z)_k \tag{13}$$

where the summation is over all the dual mesh faces that constitute the boundary of the control volume around the node *0*. The areas S_x, S_y, S_z are projections of the dual face.

The flux evaluation can be cast into edge-based operations. Consider an edge **i**, constituted by the nodes, $0, N(i)$. The quadrilateral faces of the dual mesh that are connected to the edge at its mid-point P are shown in Figure 3. The number of such quadrilateral faces connected to an edge depends on the number of cell neighbors for that edge. As an illustrative case, the edge in Figure 3 is shown to have four quadrilateral faces. The projections of the area $\bar{\mathbf{A}}_\mathbf{i}$ associated with the edge **i** are evaluated in terms of those of the quadrilateral face areas, $\bar{\mathbf{a}}_\mathbf{1}$, $\bar{\mathbf{a}}_\mathbf{2}$, $\bar{\mathbf{a}}_\mathbf{3}$, $\bar{\mathbf{a}}_\mathbf{4}$, as

$$(A_i)_x = \sum_{j=1}^{4} (a_j^*)_x, \quad (A_i)_y = \sum_{j=1}^{4} (a_j)_y, \quad (A_i)_z = \sum_{j=1}^{4} (a_j)_z \tag{14}$$

The projections are computed such that the area vector always points outward from the control volume surface associated with any node. The boundary of the control-volume around the node *0* is constituted by the union of such areas $\bar{\mathbf{A}}_\mathbf{i}$ associated with each edge **i** that share the node *0*. Thus, the summation over the dual mesh faces in equation (13) is equivalent to a summation over the edges of the grid and the fluxes are evaluated on the dual mesh faces associated with each edge. This eliminates a significant amount of computational work as the number of edges is much less than the number of faces in an unstructured grid.

In order to evaluate the contribution of each edge to the flux across the control-volume faces of a node *0*, the flux vectors **F**,**G**,**H** on the mid-point P of each edge **i** are obtained by taking the average of the

flux vectors evaluated at the nodes $0, N(i)$ on either ends of the edge using the known state variables at these nodes. This strategy has been shown to be equivalent to a finite-element Galerkin approximation which is second-order accurate in space Thus, the contribution of the edge i to the fluxes across the faces of the control volume surrounding the node 0 is given by:

$$\mathbf{F}_p (A_i)_x + \mathbf{G}_p (A_i)_y + \mathbf{H}_p (A_i)_z \tag{15}$$

The fluxes are thus evaluated on an edge-wise basis and conservation is enforced by producing a positive flux contribution to one node and an equally opposite contribution to the other node that constitutes the edge.

Similarly, the surface integral in equation (11), which is used to evaluate the second-order temporal derivative by the Lax-Wendroff approach, can be expressed in the discrete form as:

$$-\frac{1}{\Omega_0}\int_{\partial\Omega_0}\left(\tilde{\mathbf{A}}n_x + \tilde{\mathbf{B}}n_y + \tilde{\mathbf{C}}n_z\right)\frac{\partial \mathbf{U}}{\partial t}dS = -\frac{1}{\Omega_0}\sum_{k=1}^{n}\left(\tilde{\mathbf{A}}S_x + \tilde{\mathbf{B}}S_y + \tilde{\mathbf{C}}S_z\right)_k\left(\frac{\partial \mathbf{U}}{\partial t}\right)_k \tag{16}$$

where the summation is over all dual mesh faces that constitute the boundary of the control volume around the node 0. Similar to the evaluation of the first order temporal derivative, the above operations can be cast a summation over the edges.

The first order temporal derivatives evaluated at the nodes $0, N(i)$ are averaged to get the value of $\left(\frac{\partial \mathbf{U}}{\partial t}\right)$ at the mid-edge point P. The state vectors known at the nodes are averaged to get the state vector $\mathbf{U_p}$ at the

mid-edge point P with which the Jacobians $\tilde{\mathbf{A}}_i$, $\tilde{\mathbf{B}}_i$, $\tilde{\mathbf{C}}_i$, are computed.

$$A_{lm} = \left(\frac{\partial F_l}{\partial U_m}\right)\Big|_{\mathbf{U}=\mathbf{U}_p} \quad B_{lm} = \left(\frac{\partial G_l}{\partial U_m}\right)\Big|_{\mathbf{U}=\mathbf{U}_p} \quad C_{lm} = \left(\frac{\partial H_l}{\partial U_m}\right)\Big|_{\mathbf{U}=\mathbf{U}_p} \tag{17}$$

The contribution of the edge i to the evaluation of the second order temporal derivative at the node *0* is then given by,

$$\left(\tilde{\mathbf{A}}_p\left(A_i\right)_x + \tilde{\mathbf{B}}_p\left(A_i\right)_y + \tilde{\mathbf{C}}_p\left(A_i\right)_z\right)\left(\frac{\partial \mathbf{U}}{\partial t}\right)_p \tag{18}$$

The contribution, $(\delta \mathbf{U}_0^n)_c$, of the convective flux terms to the total change $\delta \mathbf{U}_0^n$ at node *0* is obtained by substituting equations (15) and (18) into the equation (4):

$$\begin{aligned}
&(\delta \mathbf{U}_0^n)_c = \\
&\quad \frac{-1}{\Omega_0}\sum_{i=1}^{n} \Delta t\left(\mathbf{F}_p\left(A_i\right)_x + \mathbf{G}_p\left(A_i\right)_y + \mathbf{H}_p\left(A_i\right)_z\right) + \\
&\quad \frac{-1}{\Omega_0}\sum_{i=1}^{n} \frac{\Delta t^2}{2}\left(\tilde{\mathbf{A}}_p\left(A_i\right)_x + \tilde{\mathbf{B}}_p\left(A_i\right)_y + \tilde{\mathbf{C}}_p\left(A_i\right)_z\right)\left(\frac{\partial \mathbf{U}}{\partial t}\right)_p
\end{aligned} \tag{19}$$

4.3 Data structure

The number of nodes in a typical tetrahedral mesh is approximately 5 to 6 times smaller than the number of cells. As a consequence, a vertex-based scheme appears to require less storage compared to a cell-centered scheme. Minimization of storage requirement is one of the main issues in development of the present scheme, which stores the state-vector values at grid-nodes. From Equation (19) it can be seen that all the operations pertaining to the evaluation of the fluxes and the dissipation terms (as will be seen in the next section) can be performed in a single loop through the edges. Hence, an edge based data structure is a natural choice for the

solver. The data structure is constituted by pointers that give the area projections associated with every edge as well as the nodes associated with the edge. As the solver is node-centered, the state vectors are stored at the nodes. These pointers provide all the information that is needed to evaluate the expressions in the relevant equations.

5 Upwind-like Artificial Dissipation

Upwind schemes for solving the hyperbolic equations in the conservation form rely on the theory of characteristics and account for the proper wave propagation directions while differencing the spatial derivatives. These schemes have shown good capability to capture the shocks without oscillations. The upwind connection to artificial dissipation in central differencing schemes is brought forth in using a simple one dimensional analogy. Adding second-difference dissipation to a second-order accurate central differencing scheme is shown to produce a first-order upwind scheme and adding a fourth-difference dissipation is shown to produce a second-order accurate upwind scheme. The motivation behind the dissipation modelling in the present work is to formulate it in such a manner as to simulate the implicit dissipation terms of the upwinding schemes, without increasing the computation cost of the algorithm.

Considering the one-dimensional Euler equation,

$$\frac{\partial \mathbf{U}}{\partial t} + \tilde{\mathbf{A}}\frac{\partial \mathbf{U}}{\partial x} = 0, \quad \tilde{\mathbf{A}} = \frac{\partial \mathbf{F}}{\partial \mathbf{U}} \tag{20}$$

given any two states $\mathbf{U_L}, \mathbf{U_R}$, the flux difference can be uniquely expressed as

$$\mathbf{F_R} - \mathbf{F_L} = \sum_k \alpha_k \lambda_k \mathbf{e}_k \tag{21}$$

where $\mathbf{e}_k$ are the right eigenvectors of $\tilde{\mathbf{A}}$. The term α_k in the summation represents the strength of each wave and λ_k represents the k^{th} eigenvalue of $\tilde{\mathbf{A}}$ (or the speed corresponding to that wave). Using this expression and accounting for the sign of the eigenvalues, the flux vector at any intermediate state **I** between **L** and **R** can be expressed as

$$\mathbf{F_I} = \frac{1}{2}(\mathbf{F_L} + \mathbf{F_R}) - \frac{1}{2}\sum_k \alpha_k \mid \lambda_k \mid \mathbf{e}_k \tag{22}$$

which can be recast into the following form

$$\mathbf{F_I} = \frac{1}{2}(\mathbf{F_L} + \mathbf{F_R}) - \tilde{\mathbf{A}}_\mathbf{r}(\mathbf{U_R} - \mathbf{U_L}) \tag{23}$$

where $\tilde{\mathbf{A}}_\mathbf{r}$ is Roe's matrix

The flux vector at the mid-edge point P was taken as the average of the flux vectors at the two nodes of the edge. This is equivalent to evaluating $\mathbf{F_I}$ using the first term in equation (23). Hence, the dissipation terms are modelled so as to be similar to the second term of the equation as this corresponds to the implicit smoothing term of the upwinding scheme. A simplified form of (23) is obtained by replacing $\tilde{\mathbf{A}}$ with $\rho\left(\tilde{\mathbf{A}}\right) = \mid u \mid + c$, the maximum eigenvalue of Roe's matrix. This ensures that the dissipation terms do not dwindle down to zero near the stagnation or the sonic points. The contribution, $(\delta \mathbf{U}_0^n)_{s2}$, of shock smoothing terms to the change $\delta \mathbf{U}_0^n$ the node *0* is given as follows:

$$(\delta \mathbf{U}_0^n)_{s2} = \frac{\Delta t}{\Omega_0}\sum_{i=1}^{n}\left(\rho(\tilde{\mathbf{A}}_\mathbf{r})S_x + \rho(\tilde{\mathbf{B}}_\mathbf{r})S_y + \rho(\tilde{\mathbf{C}}_\mathbf{r})S_z\right)_i\left(\mathbf{U_{N(i)}} - \mathbf{U_0}\right) \tag{24}$$

where $\rho(\tilde{\mathbf{A}}_\mathbf{r}), \rho(\tilde{\mathbf{B}}_\mathbf{r}), \rho(\tilde{\mathbf{C}}_\mathbf{r})$ are the spectral radii of Roe's matrices corresponding to the flux vectors **F**, **G**, **H** evaluated for the edge i. The eigenvalues are evaluated at the Roe averaged quantities The shock

smoothing term is evaluated similar to the convective fluxes on an edge-wise basis.

The background smoothing terms are modelled in a similar fashion. Instead of the first difference of state vectors as used in equation (24), a difference of the accumulated first difference over the edges sharing a node is taken. This is in concert with the one-dimensional analogy wherein such a difference is equivalent to the fourth-difference operator at the nodes. The contribution, $(\delta \mathbf{U}_0^n)_{s4}$, of background smoothing terms to the change $\delta \mathbf{U}_0^n$ at node 0 is given as follows:

$$(\delta \mathbf{U}_0^n)_{s4} = \frac{-\Delta t}{\Omega_0} \sum_{i=1}^{n} \left(\rho(\tilde{\mathbf{A}}_\mathbf{r}) S_x + \rho(\tilde{\mathbf{B}}_\mathbf{r}) S_y + \rho(\tilde{\mathbf{C}}_\mathbf{r}) S_z \right)_i \left(\delta \mathbf{U}_{\mathbf{N(i)}} - \delta \mathbf{U}_\mathbf{0} \right) \tag{25}$$

where

$$\delta \mathbf{U}_0 = \sum_k \left(\mathbf{U}_{\mathbf{N(k)}} - \mathbf{U}_\mathbf{0} \right) \tag{26}$$

is the accumulated first difference over all the edges k sharing the node 0. The background smoothing terms are evaluated on an edge-wise basis as well.

The total change $\delta \mathbf{U}_\mathbf{0}^\mathbf{n}$ at the node 0 is given by:

$$\delta \mathbf{U}_\mathbf{0}^\mathbf{n} = (\delta \mathbf{U}_0^n)_c + \sigma_2 (\Delta P_0) (\delta \mathbf{U}_0^n)_{s2} + \sigma_4 (1 - \Delta P_0) (\delta \mathbf{U}_0^n)_{s4} \tag{27}$$

ΔP is the pressure switch that is used to turn the shock smoothing and the background smoothing on at the appropriate regions. For any node 0, the pressure switch is computed as

$$(\Delta P_0) = \frac{\sum_{i=1}^{n} \left(P_{N(i)} - P_0 \right)}{\sum_{i=1}^{n} \left(P_{N(i)} + P_0 \right)} \tag{28}$$

the summation is over all the edges that share the node *0*. The pressure switch is normalized by the maximum value over the domain so that $0 \leq \Delta P \leq 1$. When evaluated as above, ΔP has a value close to zero in the smooth regions of the flow and has a value close to unity near the regions of flow discontinuities that are characterized by a pressure jump. The coefficient σ_2 is an empirical parameter that controls the amount of shock smoothing. The shock smoothing is turned on in the regions of flow that have a sharp variation in flow parameters such as near the shocks and is turned off elsewhere. The coefficient σ_4 is an empirical parameter that controls background smoothing. The background smoothing is turned on in the smooth regions of the flow and is turned off near the shock regimes using the pressure switch.

6 Local time stepping

The solution at each node is advanced in time using local time steps. A CFL stability limitation is applied for the convective terms. The viscous-like smoothing term can have appreciable magnitude at shock regions. As a consequence, a stability limitation that combines both the inviscid, and the diffusion limitations is applied. The time-step restriction for the 1-D wave equation is $\Delta t \leq \frac{\Delta x}{|u|+c}$, while the restriction for the 1-D diffusion equation is $\Delta t \leq \frac{1}{2}\frac{\Delta x^2}{\nu}$, where in this case $\nu = \sigma_2 \Delta P$. The formula for the time-step, Δt_0, for any node *0* then is given by:

$$\Delta t_0 = \omega \frac{V_0}{A_x + A_y + A_z + D}, \tag{29}$$

where

$$A_x = (|u_0| + \alpha_0) S_{x0}$$

$$A_y = (|v_0| + \alpha_0) S_{y0}$$

$$A_z = (|w_0| + \alpha_0) S_{z0},$$

and

$$D = 2\sigma_2 \Delta P_0 \frac{V_0}{S_{x0} + S_{y0} + S_{z0}}. \tag{30}$$

In the above expression u_0, v_0, w_0 are the velocity components at the node *0*, α_0 is the speed of sound, and ΔP is the pressure switch. The area terms S_{x0}, S_{y0}, S_{z0} are the projected areas of the dual volume around the node *0* in the x,y, and z directions, and are given by the formulas:

$$S_{x0} = \frac{1}{2} \sum_{j=1}^{n} |S_x|_j \ , S_{y0} = \frac{1}{2} \sum_{j=1}^{n} |S_y|_j \ , S_{z0} = \frac{1}{2} \sum_{j=1}^{n} |S_z|_j \tag{31}$$

where the summation is over all the dual mesh faces that constitute the boundary of the control volume around the node *0*. A value of the factor $\omega = 0.5$ has been employed.

7 Boundary Conditions

Three types of conditions have been applied for the cases considered in the present work. Those are (i) flow tangency to wall, (ii) far field, and (iii) symmetry. The flow tangency condition is imposed by extrapolating the velocity at the center of a boundary cell to the corresponding boundary face, and then subtracting the component normal to the solid surface. Density and pressure values are extrapolated from the cell-center to the boundary face. Characteristic boundary conditions are applied at the far field boundaries. The characteristic variables that are employed are the Riemann invariants, the entropy, and the two velocity components that are tangent to the boundary. Lastly, the symmetry condition

is applied by simply extrapolating the state-vector from the cell-center to the corresponding boundary face.

8 Adaptive Grid Refinement

A dynamic grid adaptation algorithm has been previously developed for 3D unstructured grids. The algorithm is capable of simultaneously un-refining and refining the appropriate regions of the flow regime, by detecting the local flow *features*. In the case of inviscid flows, the dominant flow *features* may be shock waves, expansion waves or vortices. The regions of existence of such features are not known *a priori* and they have to be detected. A feature detector senses the flow *features* that are present in different regions and guides the adaptive algorithm to embed these regions if the existing grid spacing in such regions is not sufficient for resolving the local flow variations. The detector visits the regions of the grid that were embedded at earlier passes of grid adaptation and checks if strong flow *features* are still prevailing in these regions. If the features have moved away from these regions, as is common in unsteady flow situations, the detector guides the adaptive algorithm to unrefine the grid locally in those regions of the grid.

8.1 Feature detection

The feature detector uses velocity differences and velocity gradients across the edges as the parameters for sensing the flow *features*. The threshold values for the parameters are set based on the distribution of the parameters which is characterized by the average (S_{ave}) and the standard deviation (S_{sd}) of the respective parameters, where S is any

detection parameter In the present work, velocity differences and velocity gradients are used as the detection parameters. The following relations are used to set the threshold parameters for refinement.

$$S_{ref.th} = S_{ave} + \alpha S_{sd}$$

The average and the standard deviation are defined as:

$$S_{ave} = \frac{\sum_{i=1}^{nedges} S_i}{nedges} \qquad S_{sd} = \sqrt{\frac{\sum_{i=1}^{nedges} S_i^2}{nedges}}$$

The value of the parameter α is chosen empirically, typical value of the parameter being 0.3. The edges that have a detection parameter value greater than the threshold value are flagged to be refined. Following the edge flagging, cells that are having four or more of their edges flagged to be divided are marked for refinement.

8.2 Cell division strategies

The different strategies usually employed for embedding a tetrahedron and the methods of cell division employed in the adaptive algorithm used in the present work are discussed

Octree division

The tetrahedra that are flagged for refinement are embedded by the octree cell-division that divides the *parent* cell into eight *children*, as shown in Figure 4, by inserting mid-edge nodes on the parent cell edges. The four corner child cells are similar to the parent cell. The four interior child cells are formed by dividing the interior octahedron, constituted by the nodes 5-6-7-8-9-10, by the shortest diagonal.

After the octree division of a portion of the grid cells, the resulting grid contains a number of cells that were initially not flagged for division but eventually are left with mid-edge nodes on some or all of their six edges due to refinement in the neighboring cells. These are termed as the *interface cells* as they constitute the border between the divided and the undivided cells and their mid-edge nodes are termed as *hanging nodes.* Numerical schemes usually employ normal tetrahedral cells with four corner nodes and significant changes are necessary in order for the schemes to be applied to such cells with additional *hanging nodes.* This is not desired, as then the adaptive algorithm becomes dependent on the specific numerical scheme that is employed. Hence, a special method of cell division has been incorporated in the adaptive algorithm which eliminates such *interface* cells. There are different configurations in which these hanging nodes appear in the interface cells.

Directional division

In the case in which all the hanging nodes are appearing on the edges of the same face, the interface cell is directionally divided into four children as shown in Figure 5. There are four possible cases of such division depending on which one of the faces the hanging nodes appear. If there is a hanging node appearing on only one of the six edges, the interface cell is henceforth divided into two children as shown in Figure 6. Depending on the edge that has the hanging node, there are six possible cases of such a division.

Centroidal node division

If the *hanging node* configuration is any different from the ones discussed above, the interface cell is treated by introducing a centroidal node and dividing the cell accordingly, as illustrated in Figure 7. It shows a case of a cell with two hanging nodes, on edges 1-3 and 2-4. A node C is introduced at the centroid of the cell and connected to the corner nodes as well as to the *hanging* nodes, thus forming tetrahedral child cells. This approach is general enough to handle the different hanging node configurations that arise.

9 Numerical Results

Two flow cases are employed in order to provide an assessment of accuracy, robustness, and computer requirements of the developed Euler solver on adaptive tetrahedral grids. The first case is transonic inviscid flow around the ONERA M6 wing. Two different initial grids have been used to obtain the flow solutions on this configuration. The second case considers the Low-Wing Transport (LWT) aircraft. The initial grids used for all the computations have been generated by an advancing front grid generation method

All the computations were performed on a CRAY Y-MP. The code was vectorized to run at a speed of about 100 Mflops. The memory required for the solver was 30 words/node. It should be noted that this memory requirement is quite small for an unstructured grid Euler solver. The values of the smoothing coefficients have been $\sigma_2 = 10^{-2}$ for shock-capturing, and $\sigma_4 = 10^{-3}$ for background smoothing.

9.1 ONERA M6 Wing

The ONERA M6 Wing is considered for evaluating accuracy of adaptive flow solutions. This configuration has been used as a benchmark case for evaluating the accuracy of several Euler methods. The wing has a leading edge sweep of 30 degrees, an aspect ratio of 3.8, a taper ratio of 0.56 and symmetrical airfoil sections. The wing has a root chord of 0.67 and a semi span of 1.0 with a rounded tip. The computational domain is bounded by a rectangular box with boundaries at $-6.5 \leq x \leq 11.0, 0.0 \leq y \leq 2.5$ and $-6.5 \leq z \leq 6.5$. Inviscid, transonic flow solutions were computed at $M_\infty = 0.84$, and angle of attack $\alpha = 3.06°$.

Adaptive solution with an initial fine grid

The initial mesh employed comprises of 231507 cells and 42410 nodes. The triangulation on the wing surface is shown in Figure 8. The solutions are started from the freestream conditions being specified everywhere. Figure 9 shows the flow solution after 4600 iterations on the initial grid. Mach number contour lines on the upper surface of the wing are shown, plotted using an increment of $\Delta M = 0.02$. The solution clearly features a λ shock that is formed by the two in-board shocks which merge together to form a single strong shock in the tip region of the wing. The fore shock is captured reasonably well whereas the aft shock appears to be more diffused. This is due to resolution being less in that region of the grid. The convergence history for the solution obtained on the initial grid is shown in Figure 10.

The pressure coefficients on the surface of the wing are compared

with the experimental results at three different spanwise locations along the wing, namely, $\eta = 0.44, \eta = 0.65, \eta = 0.90$. (Figures 11 to 13). The Euler result obtained on the initial grid is shown with a dashed line. Experimental data for the pressure coefficients on the upper surface are represented by the filled circles, while the data for the lower surface are represented by the unfilled circles. At station $\eta = 0.44$, there are two shocks observed along he chord. It is observed that the pressure jumps of both the fore and the aft shocks are falling short of the experimental observations. At station $\eta = 0.65$, it is seen that the two shocks are closer to each other. The computed pressure jump across the fore shock is smaller than the experimental values and the aft shock is quite smeared. At station $\eta = 0.90$ the two shocks have merged together to form a single strong shock. From the figure, it is seen that the scheme captures the shock fairly well at this location. However, the leading edge suction peak is under-predicted.

The initial grid is now adapt[illegible] embedded in the regions of the local flow features. Figures 14 and 15 show the triangulation on the wing surface and the symmetry planes with the embedded regions of the grid being denoted with the darker shades. Velocity differences and the velocity gradients were used as the detection parameters. The adapted grid has 833613 cells and 144722 nodes. It is seen that grid embedding is aligned along the λ shock. Furthermore, there is more embedding along the aft shock than along the fore shock as the former is more smeared in the initial solution. There is also an appreciable amount of embedding in the leading edge region of the wing as the flow undergoes rapid acceleration from the stagnation point and reaches the peak Mach

number of about 1.50 within 10% chord at all span wise locations. There is also some embedding on the symmetry plane near the leading edge region and near 75% chord which shows the presence of the weak aft shock at the latter location.

The solution obtained on the initial grid is interpolated to the new grid points and this is used as the starting solution for the adapted grid. Figures 16 and 17 show the solution obtained on the adapted grid. Mach number contour lines on the upper surface of the wing and on the symmetry plane are shown, plotted using an increment of $\Delta M = 0.02$. It is seen from the figures that both the fore and the aft shocks have sharpened to an appreciable extent compared to the corresponding solution plots on the initial grid (Figure 9). It is also observed that the aft shock on the symmetry plane is much sharper. The pressure coefficients comparison of the adapted grid solution with the experimental observations is shown in Figures 11 to 13, at three spanwise locations. The pressure coefficient distribution corresponding to the adapted grid solution is shown in solid lines. It is seen that at all the spanwise locations, the agreement of the computed solution with the experimental observations has improved considerably from that of the initial grid solution (dashed lines). The leading edge suction peak is captured well at all the locations. It is observed that at station $\eta = 0.44$, the aft shock location in the final solution is downstream of the experimentally measured location. This discrepancy has been observed in solutions with other Euler schemes The pressure coefficient distribution on the lower surface of the wing is in good agreement with the experimental results.

9.2 Low-Wing Transport Aircraft

Computations were carried out for a low wing transport (LWT) configuration Inviscid, transonic flow of $M_\infty = 0.768$ was considered with an angle of attack $\alpha = 1.116°$. A semispan computational grid was generated for the LWT aircraft geometry without the nacelles. View of the triangulation on the wing upper surface and the fuselage is shown in Figure 26. The initial grid is comprised of 48828 nodes and 266400 cells. The experimental pressure measurements were obtained with Reynolds number of 2.5×10^6 based on the mean aerodynamic chord of the wing

Computed solutions obtained on the initial grid are shown in Figure 27. Mach number contour lines are plotted at intervals of $\Delta M = 0.02$. A single shock wave is formed on the upper surface. Comparison of the pressure coefficients at three spanwise locations, namely $\eta = 0.20, \eta = 0.40$ and $\eta = 0.55$, on the wing surface are shown in Figures 28 to 30. The experimental values corresponding to the upper surface are shown in filled circles and the values corresponding to the lower surface are shown in unfilled circles. Comparing the C_p values obtained on the initial grid, shown in dashed lines in Figures 28 to 30, with the experimental data shows that the results agree quite well in the fore regions of the wing, but agreement is somewhat poor in the aft regions. This has been observed by other inviscid flow computations as well . The viscous effects change the flow pattern to a considerable extent in this case. Viscous effects have been observed to be very significant in the aft portion of the wing due to flow separation. The grid was embedded using the velocity gradients and

velocity differences as the parameters. The embedded grid, on the aircraft surface as well as on the symmetry plane, is shown in Figures 31 and 32. The adapted grid has 185230 nodes. The Mach number contour lines are shown in Figure 33, plotted at intervals of $\Delta M = 0.02$. The plot shows that there is a considerable improvement in the solution as compared to the one obtained on the initial grid. The shock appears distinctly sharper on the adapted grid solution. Comparison of the pressure coefficients, obtained using the adapted grid solution, on the wing surface with the experimental values is shown in solid lines in Figures 28 to 30. The figures show that the solution on the adapted grid captures the suction peak in the fore region of the wing better than the initial grid solution.

Figure 1: Dual Mesh, shown in dashed lines, for a Triangular Grid

Figure 2: Dual Mesh , shown in dashed lines, for a Tetrahedral Grid

Figure 3: Dual Mesh faces attached to an edge

Figure 4: Octree division of a tetrahedron

Figure 5: Directional cell division into four children when all three hanging nodes are on the same face

Figure 6: Directional cell division into two children when there is only one hanging node

Figure 7: Centroidal Node Division of an Interface Cell

Figure 8: Triangulation of the ONERA wing upper surface (initial fine grid)

Figure 9: Mach number contour lines on the wing upper surface - Solution obtained on the initial fine grid after 4600 iterations. ($\Delta M = 0.02$)

Figure 10: Convergence History for the solution obtained on the initial fine grid for the ONERA M6 wing

Figure 11: Pressure Coefficients comparison on the wing surface at $\eta = 0.44$ spanwise location. • – Experimental Values (upper surface), o – Experimental Values (lower surface), - - - Initial grid solution, ——— One level adapted grid.

Figure 12: Pressure Coefficients comparison on the wing surface at $\eta = 0.65$ spanwise location. • – Experimental Values (upper surface), o – Experimental Values (lower surface), - - - Initial grid solution, ——— One level adapted grid.

Figure 13: Pressure Coefficients comparison on the wing surface at $\eta = 0.90$ spanwise location. • – Experimental Values (upper surface), o – Experimental Values (lower surface), - - - Initial grid solution, ——— One level adapted grid.

Figure 14: Triangulation on the wing upper surface corresponding to once adapted grid

Figure 15: Isometric view of the triangulation on the wing upper surface and the symmetry plane corresponding to the once adapted fine grid.

Figure 16: Mach number contour lines on the wing upper surface - Solution obtained on the once adapted fine grid. ($\Delta M = 0.02$)

Figure 17: Mach number contour lines on the wing upper surface and symmetry plane - Solution obtained on the once adapted fine grid. ($\Delta M = 0.02$)

Figure 26: Triangulation on the LWT aircraft wing and body surface corresponding to the initial grid.

Figure 27: Mach number contour lines on the wing upper surface - Solution obtained on the initial grid. ($\Delta M = 0.02$)

Figure 28: Pressure Coefficients comparison on the wing surface at $\eta = 0.20$ span-wise location. • – Experimental Values (upper surface), o – Experimental Values (lower surface), – – – Initial grid solution, ——— One level adapted grid.

Figure 29: Pressure Coefficients comparison on the wing surface at $\eta = 0.40$ span-wise location. • – Experimental Values (upper surface), o – Experimental Values (lower surface), – – – Initial grid solution, ——— One level adapted grid.

Figure 30: Pressure Coefficients comparison on the wing surface at $\eta = 0.55$ span-wise location. • – Experimental Values (upper surface), o – Experimental Values (lower surface), – – – Initial grid solution, ——— One level adapted grid.

Figure 31: Triangulation on the wing and body surface corresponding to the once adapted grid.

Figure 32: Isometric view of the triangulation on the wing, body and symmetry plane surfaces corresponding to the once adapted grid.

Figure 33: Mach number contour lines on the wing upper surface - Solution obtained on the once adapted grid. ($\Delta M = 0.02$)

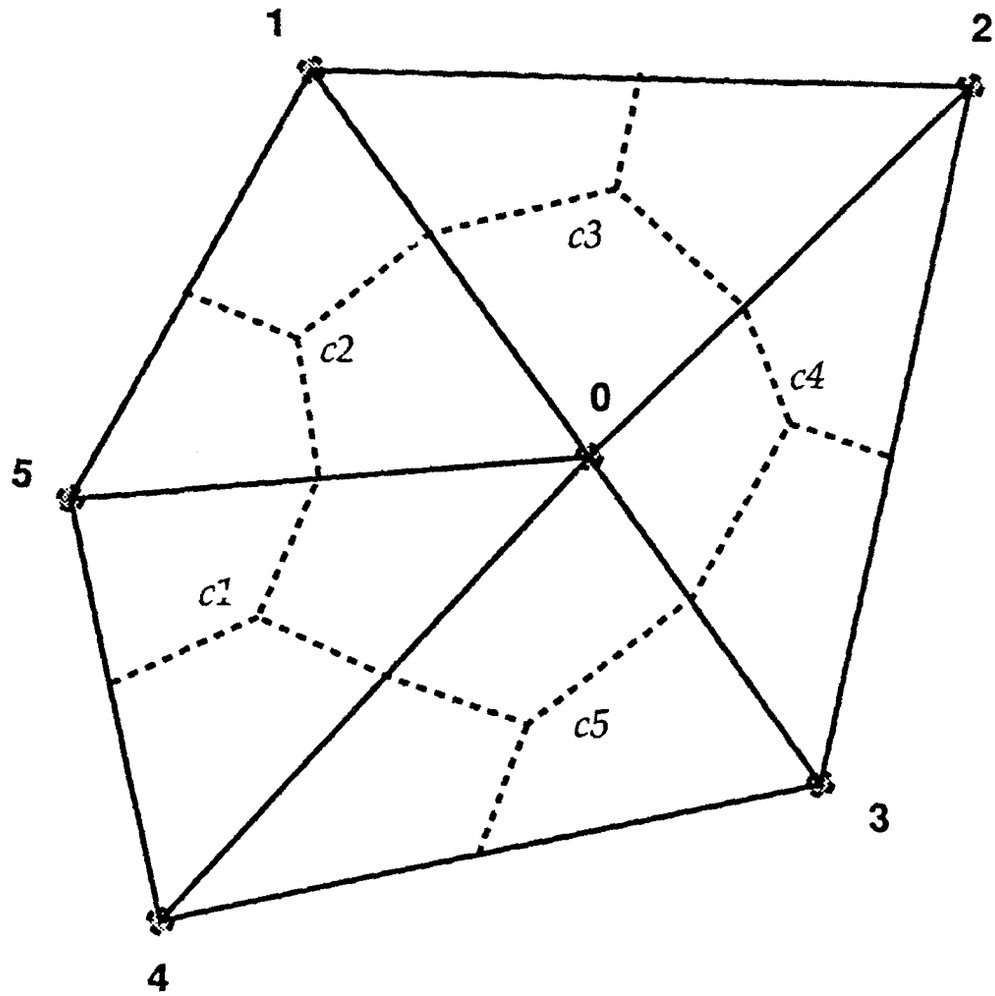


Fig 1

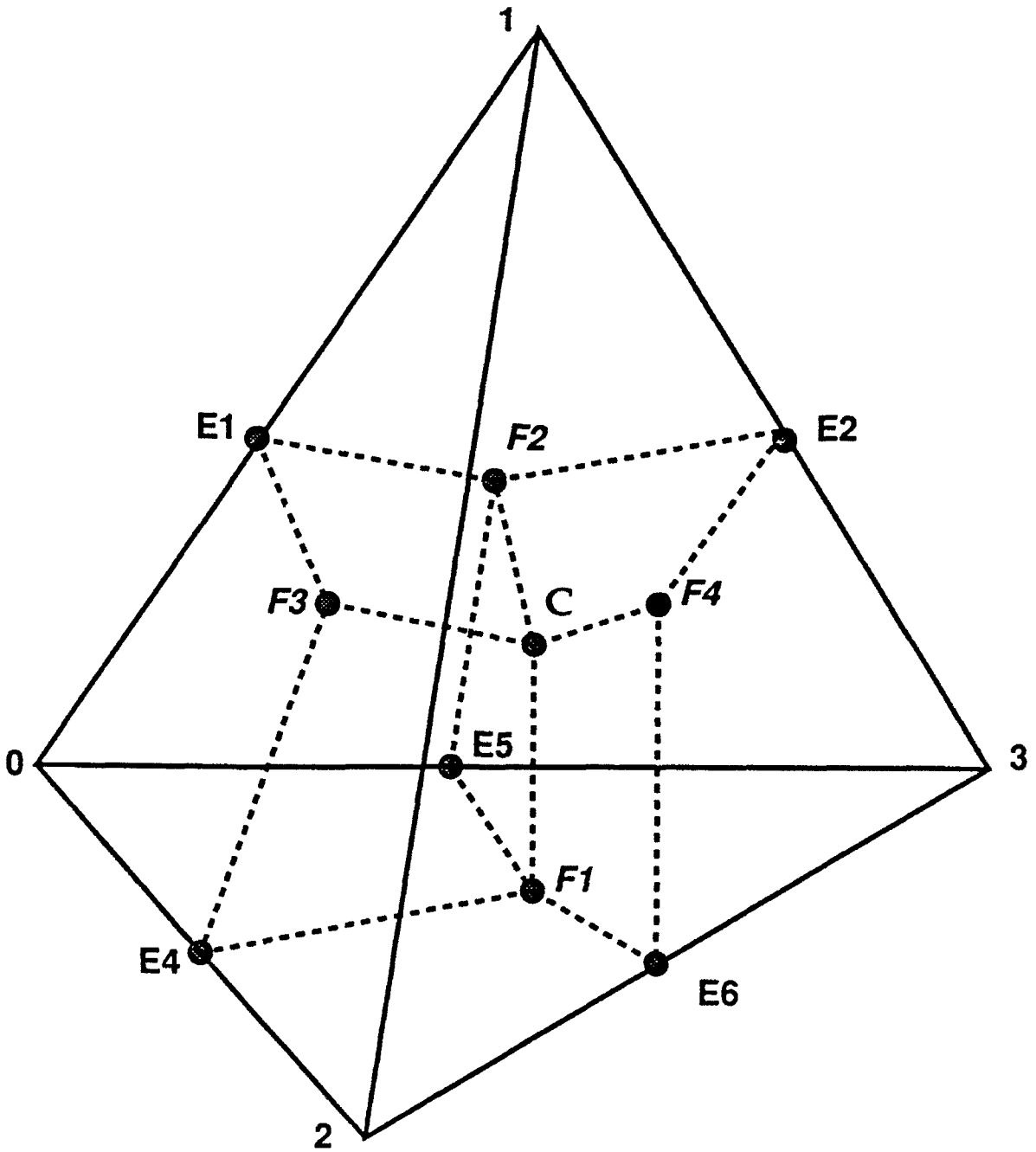
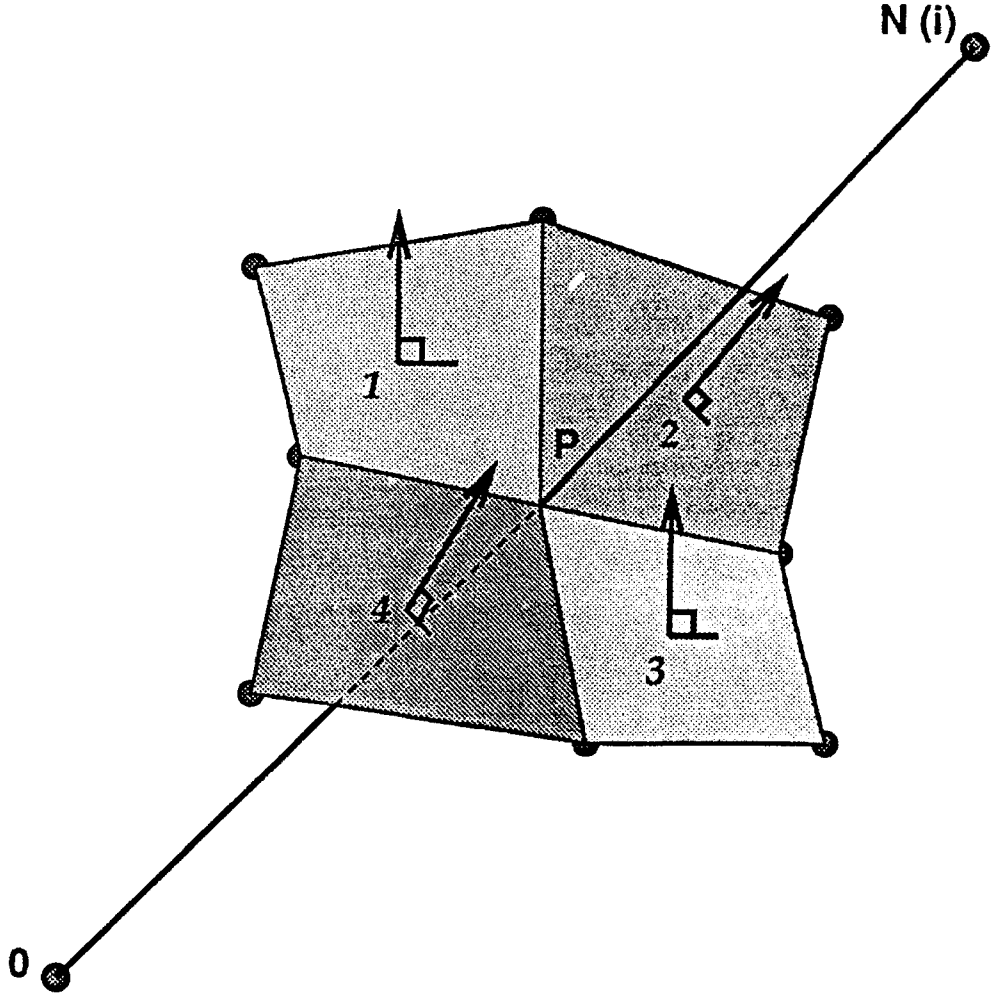
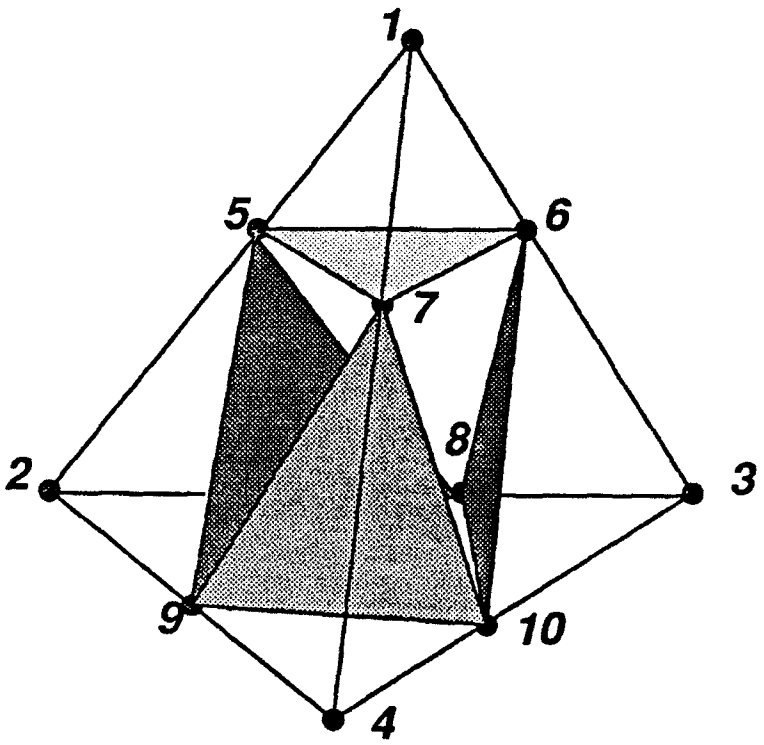


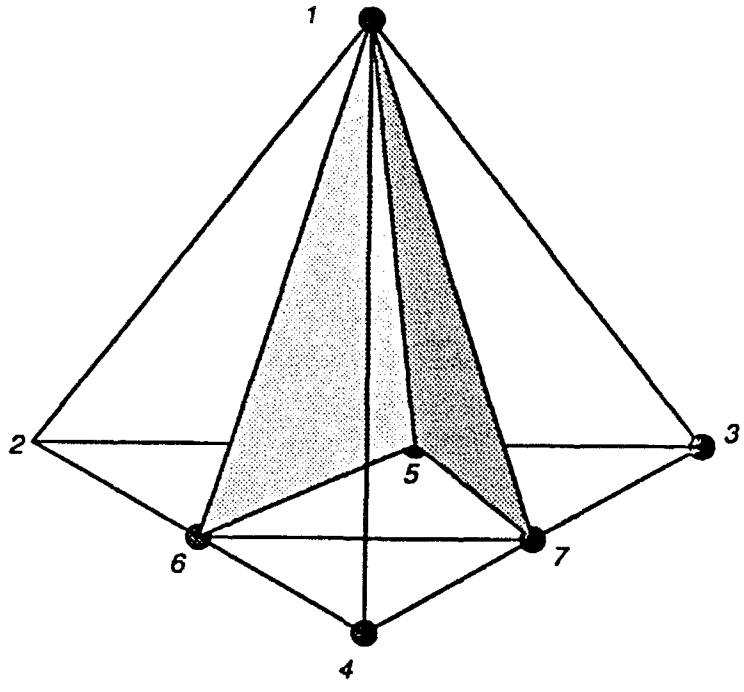
Fig 2


N (i)
1
P
2
4
3
0

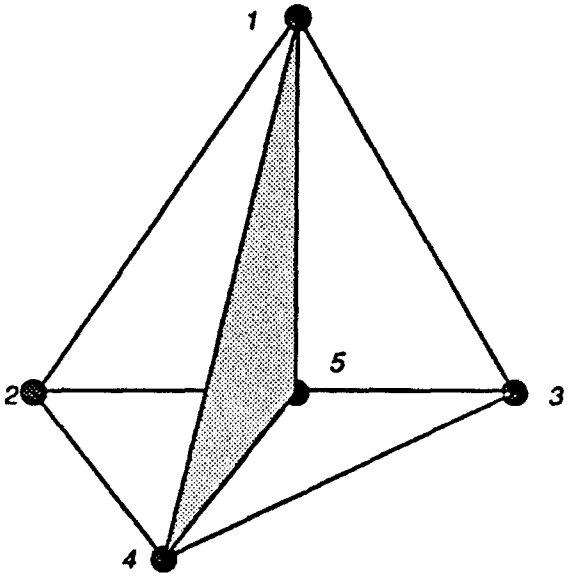


CORNER CHILD CELLS *(nodes 1,5,6,7 2,5,8,9 3,6,8,10 4,7,9,10)*

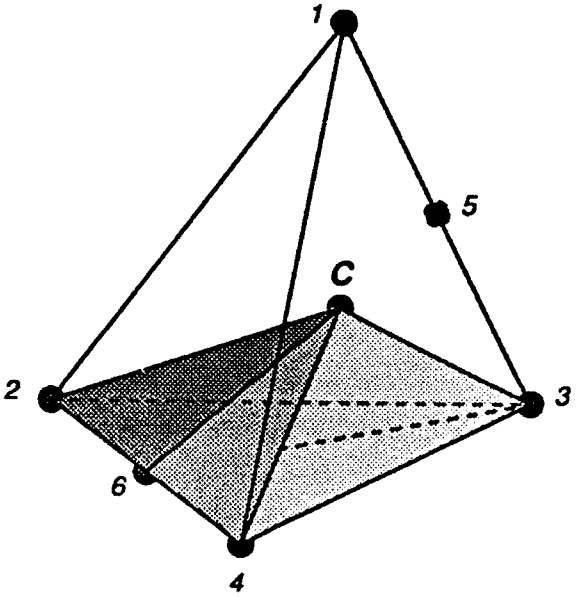
INTERIOR CHILD CELLS *(nodes 5,6,7,8 5,7,8,9, 6,7,8,10 7,8,9,10)*



FOUR CHILD CELLS (nodes 1,2,5,6 1,3,5,7 1,4,6,7 1,5,6,7)

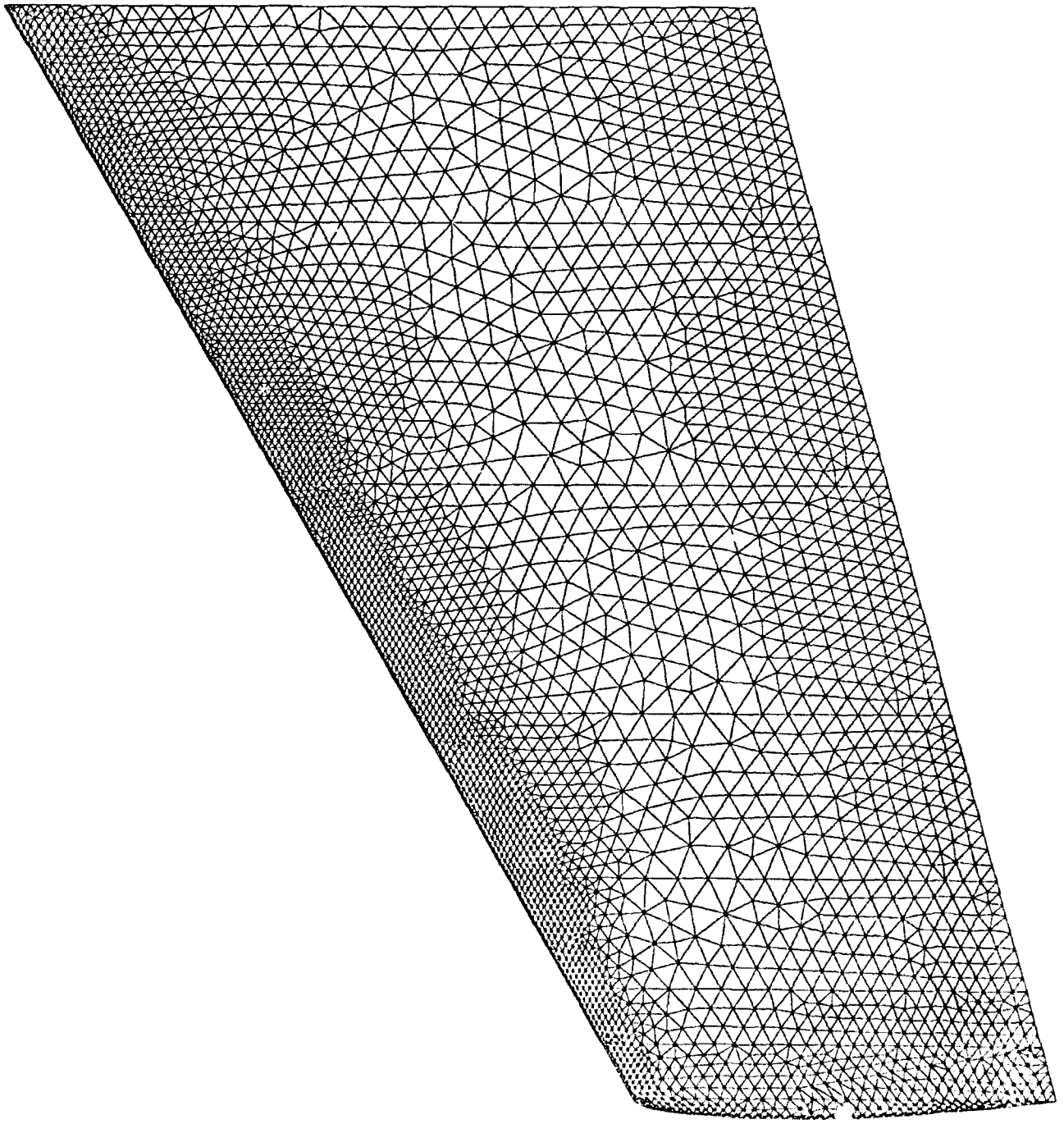


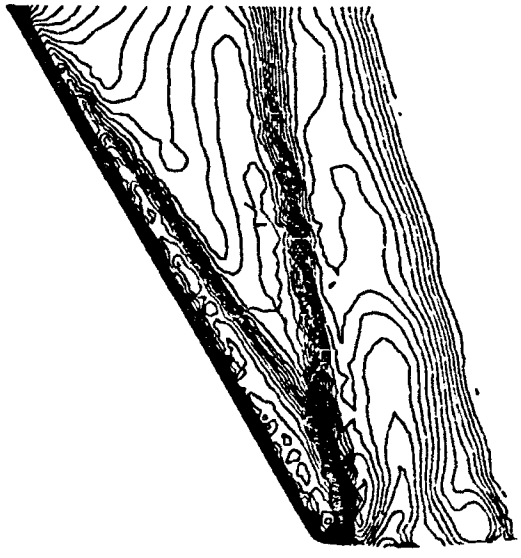
TWO CHILD CELLS (nodes 1,2,4,5 1,3,4,5)



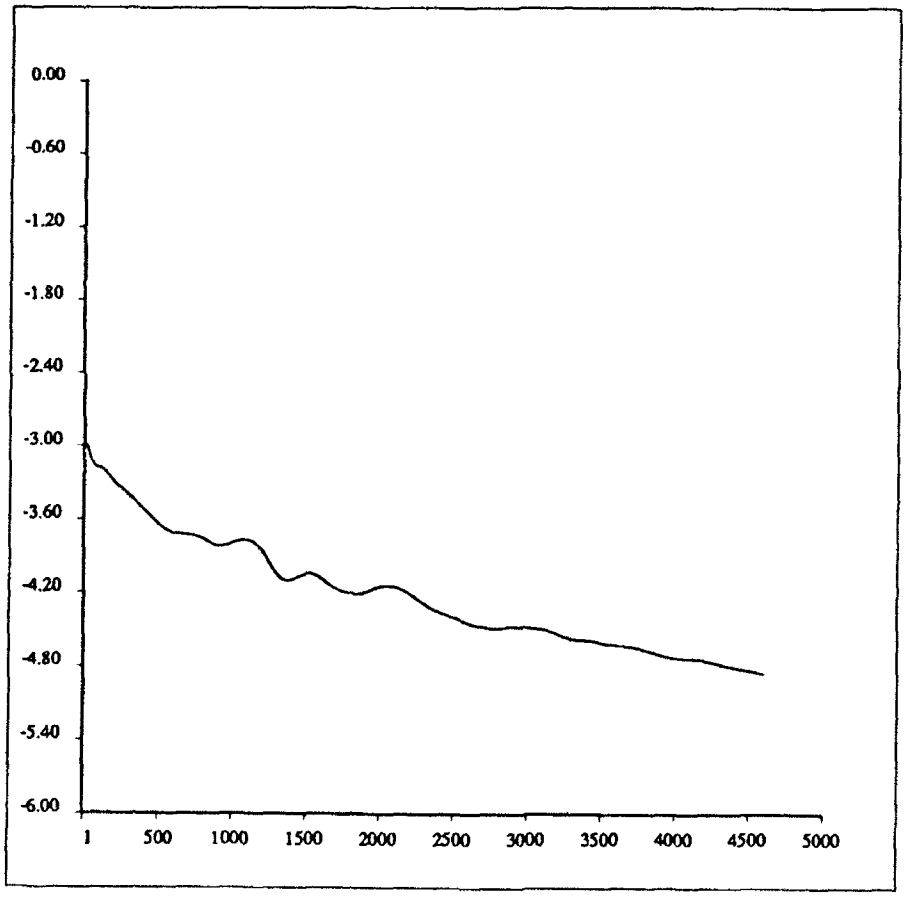
CHILDREN CELLS ON FACE 2,3,4

- *nodes C,3,4,6*
- *nodes C,2,3,6*



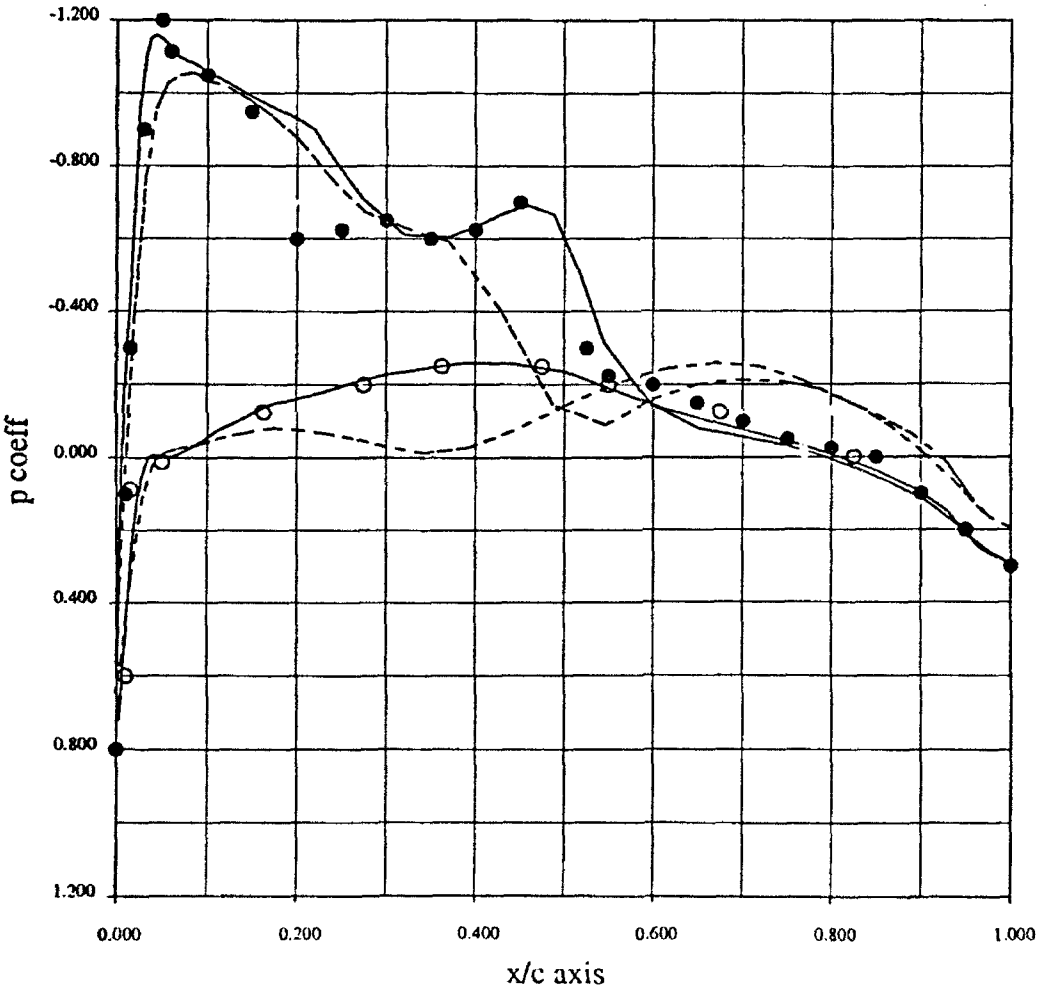


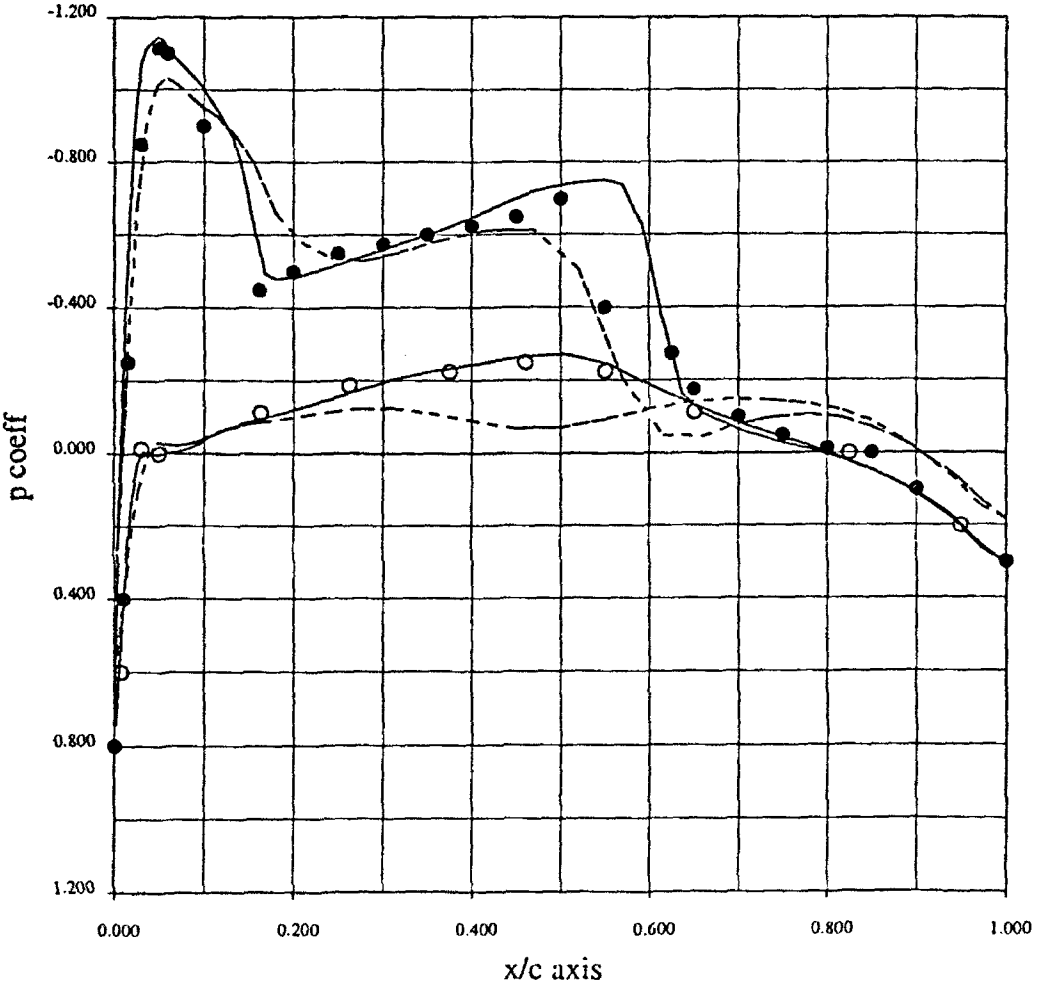
Residual History

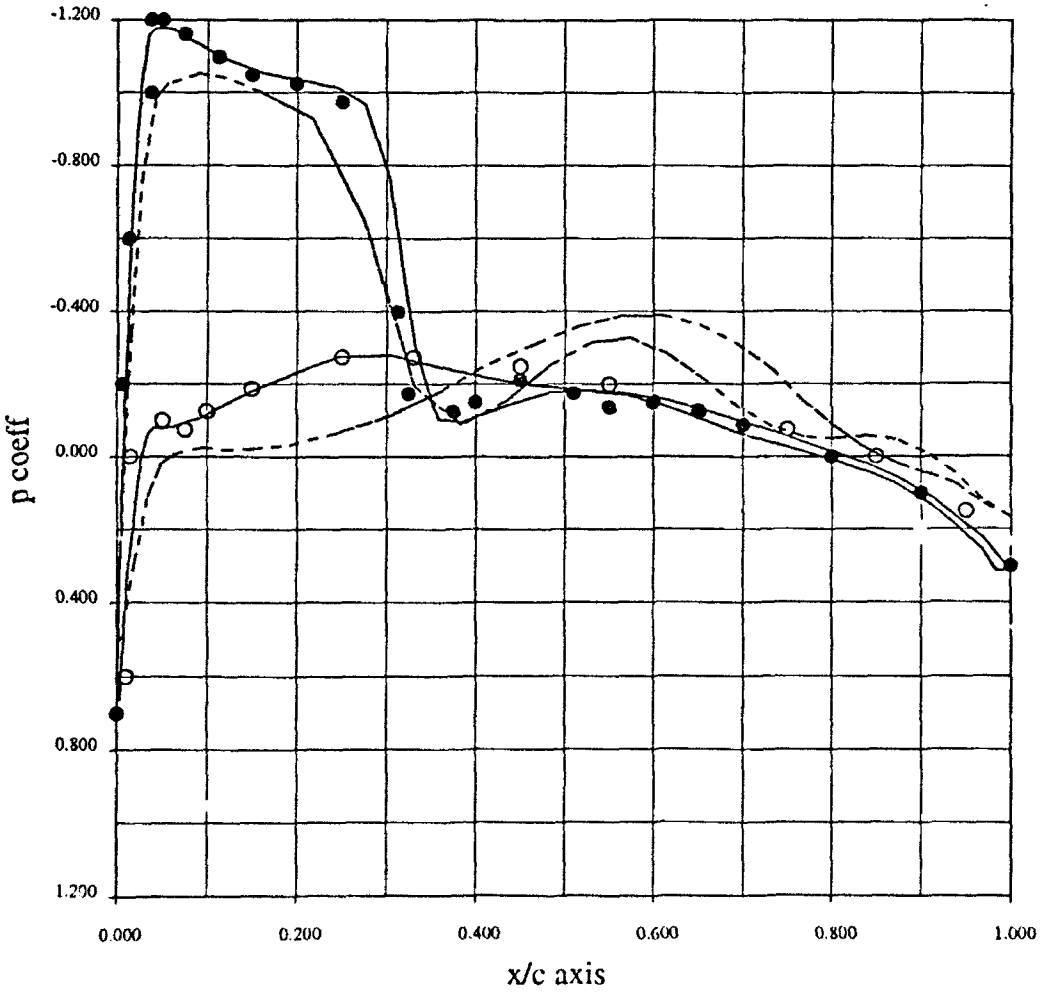


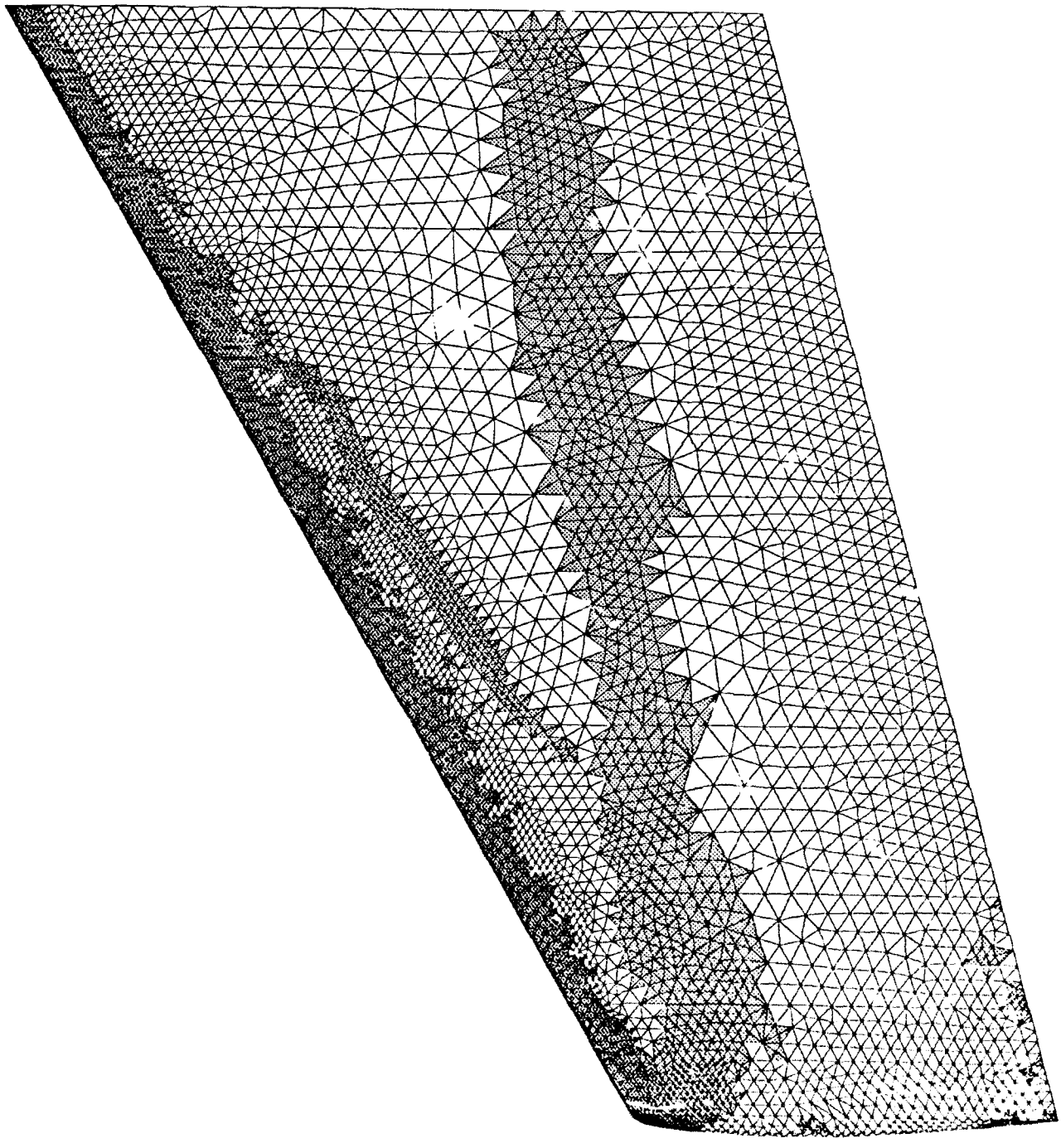
Log (RMS residual)

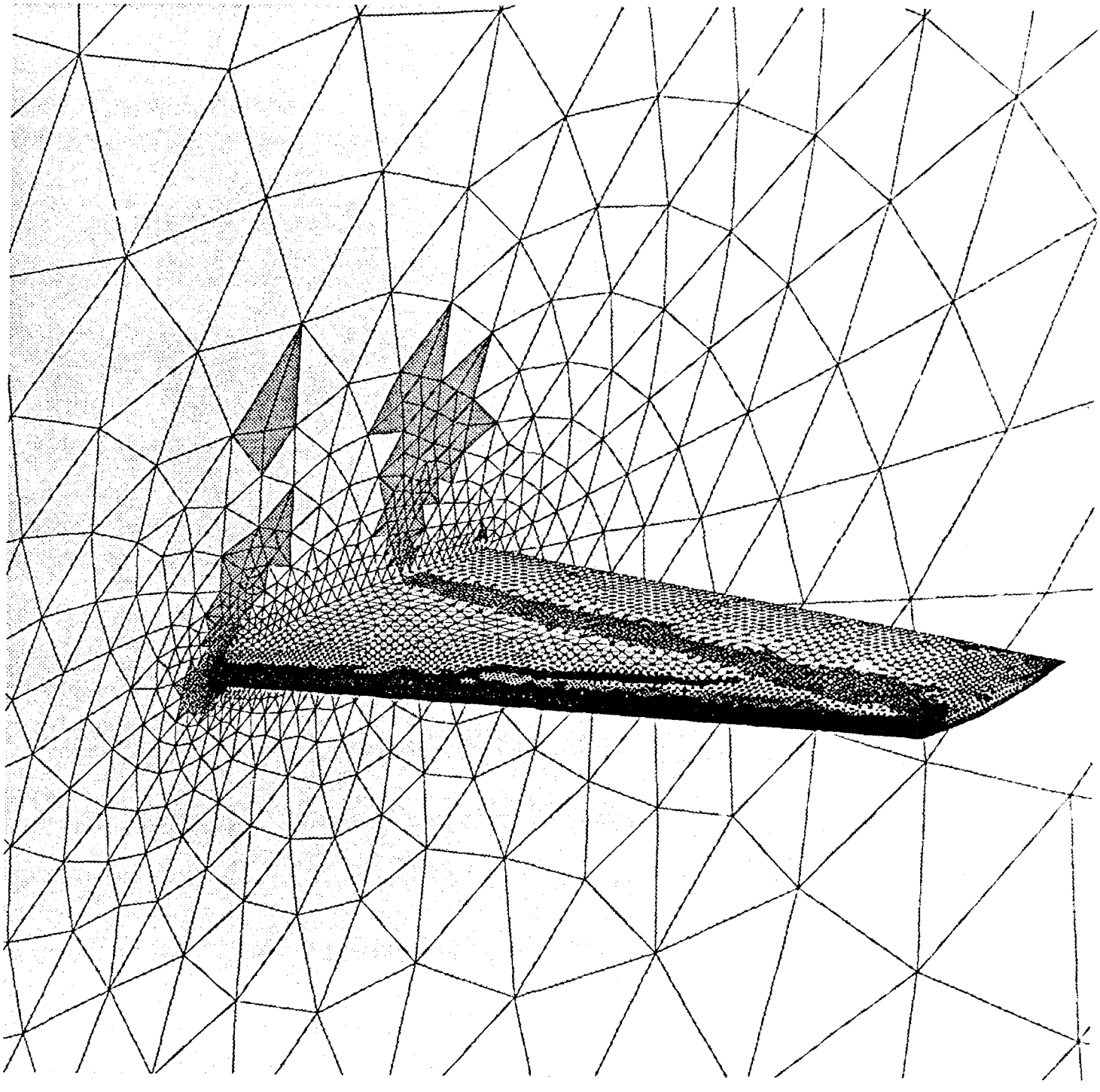
Number of Iterations

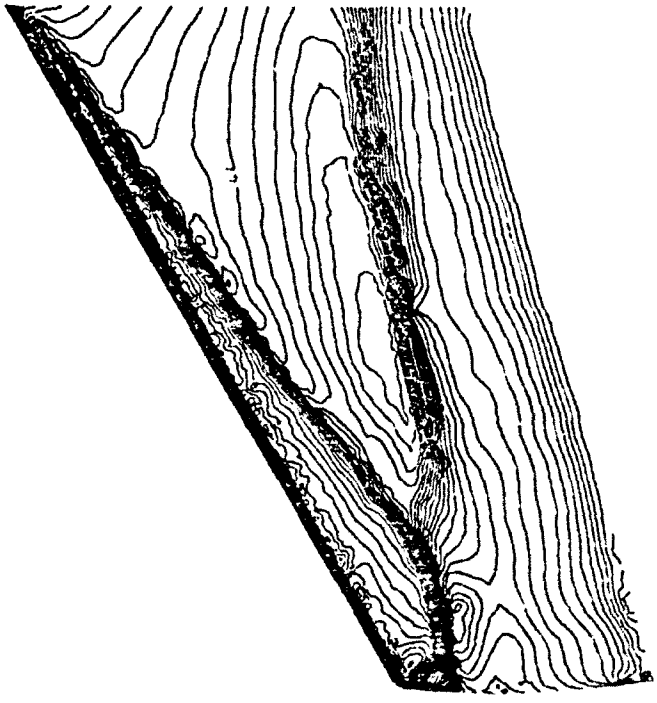

-1.200
-0.800
-0.400
0.000
0.400
0.800
1.200
p coeff
0.000
0.200
0.400
0.600
0.800
1.000
x/c axis

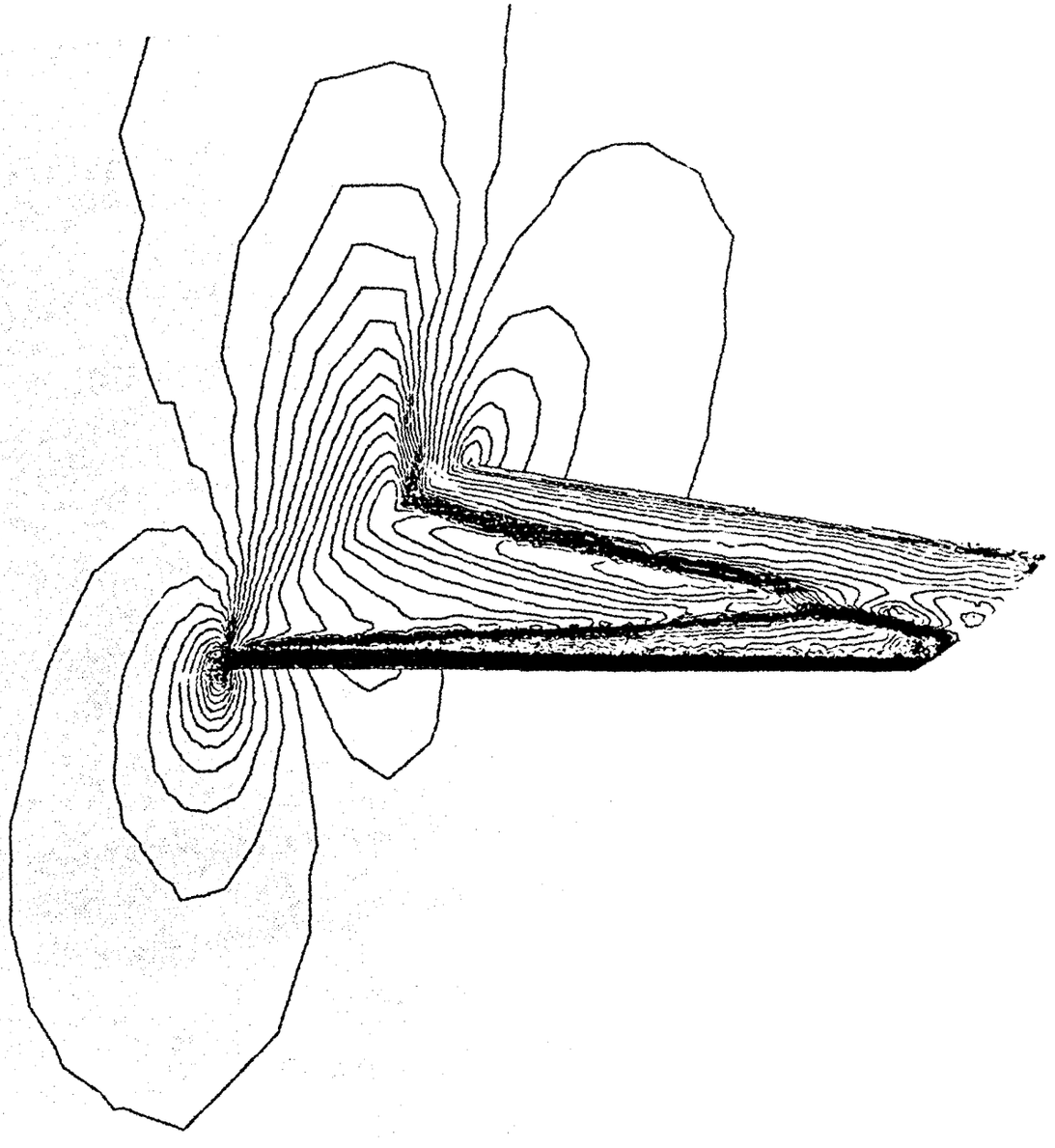

-1.200
-0.800
-0.400
0.000
0.400
0.800
1.200
p coeff
0.000
0.200
0.400
0.600
0.800
1.000
x/c axis

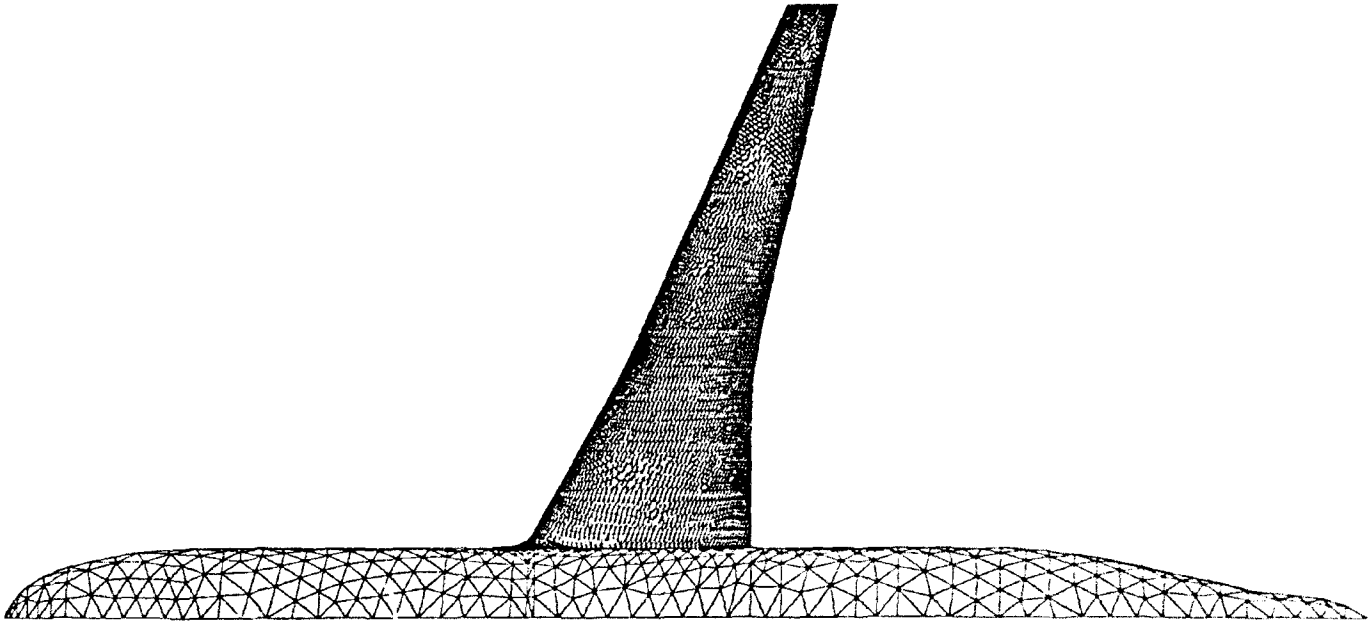

-1.200
-0.800
-0.400
0.000
0.400
0.800
1.200
p coeff
0.000
0.200
0.400
0.600
0.800
1.000
x/c axis

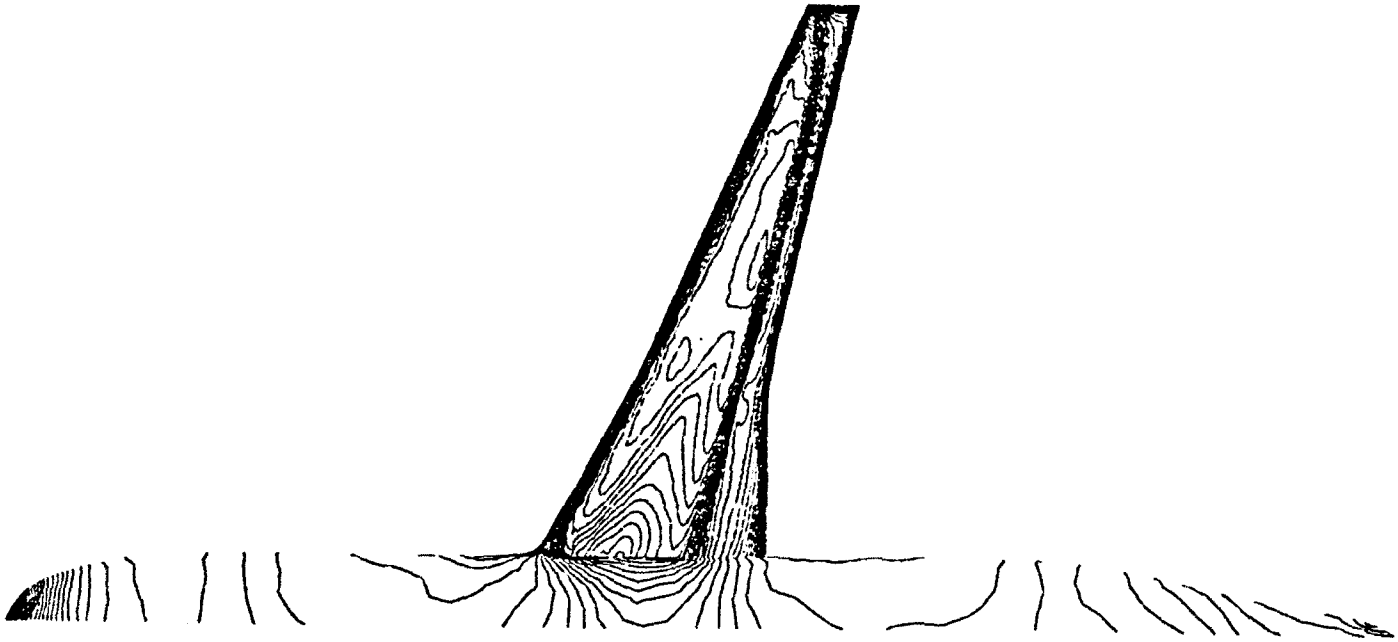


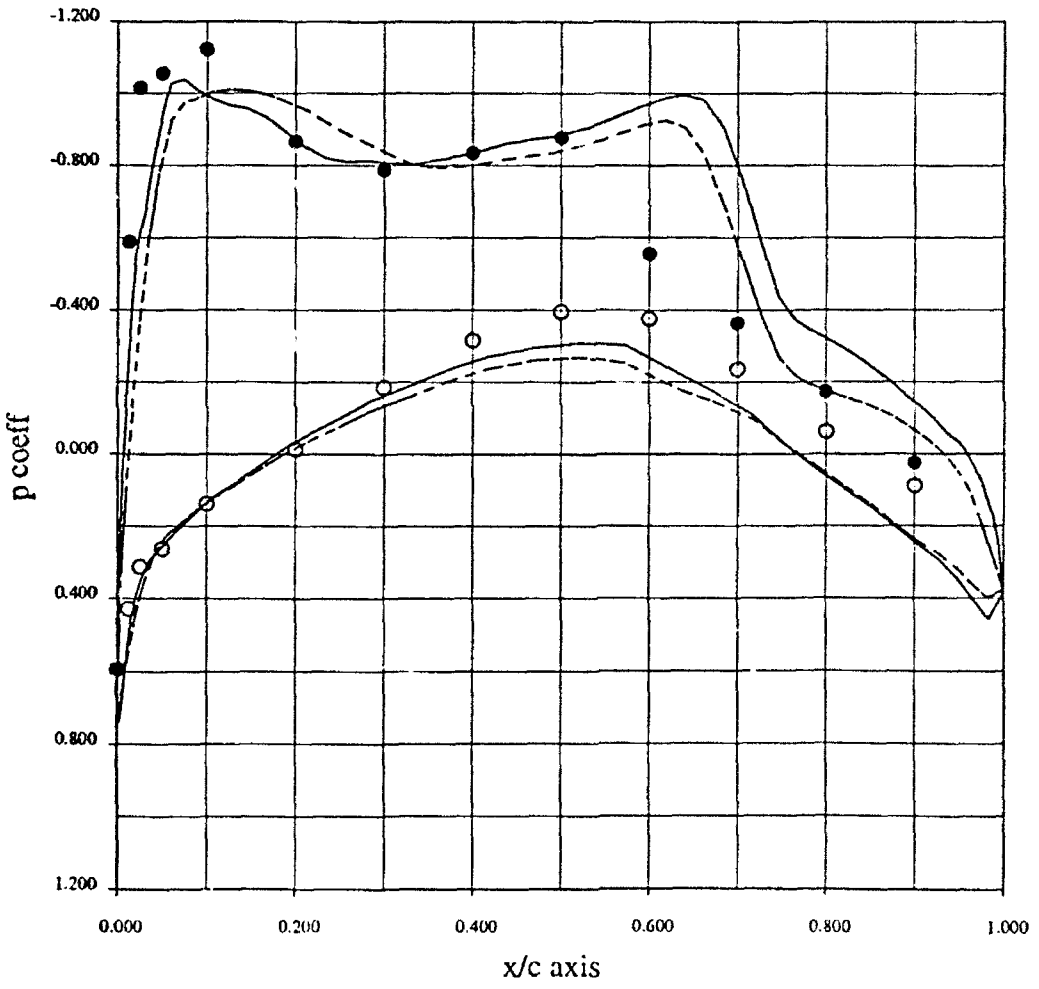


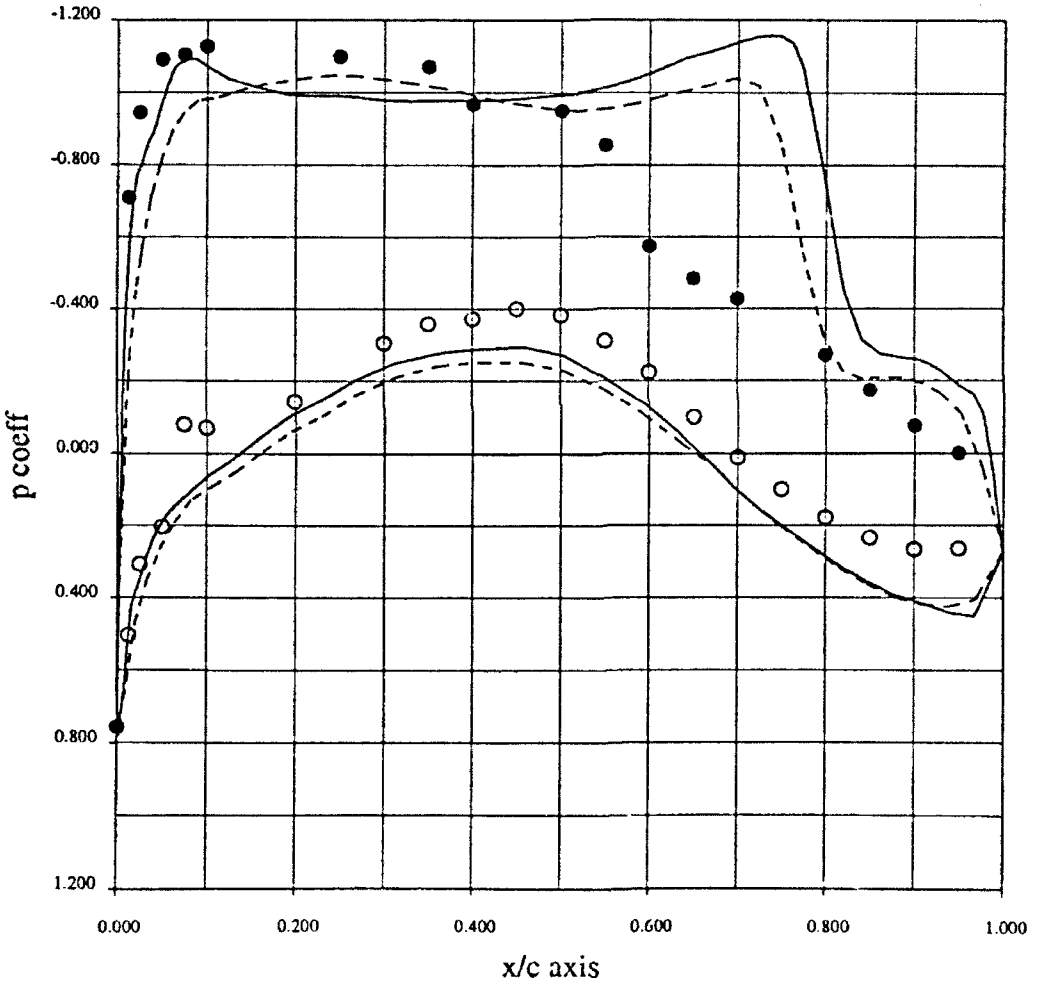


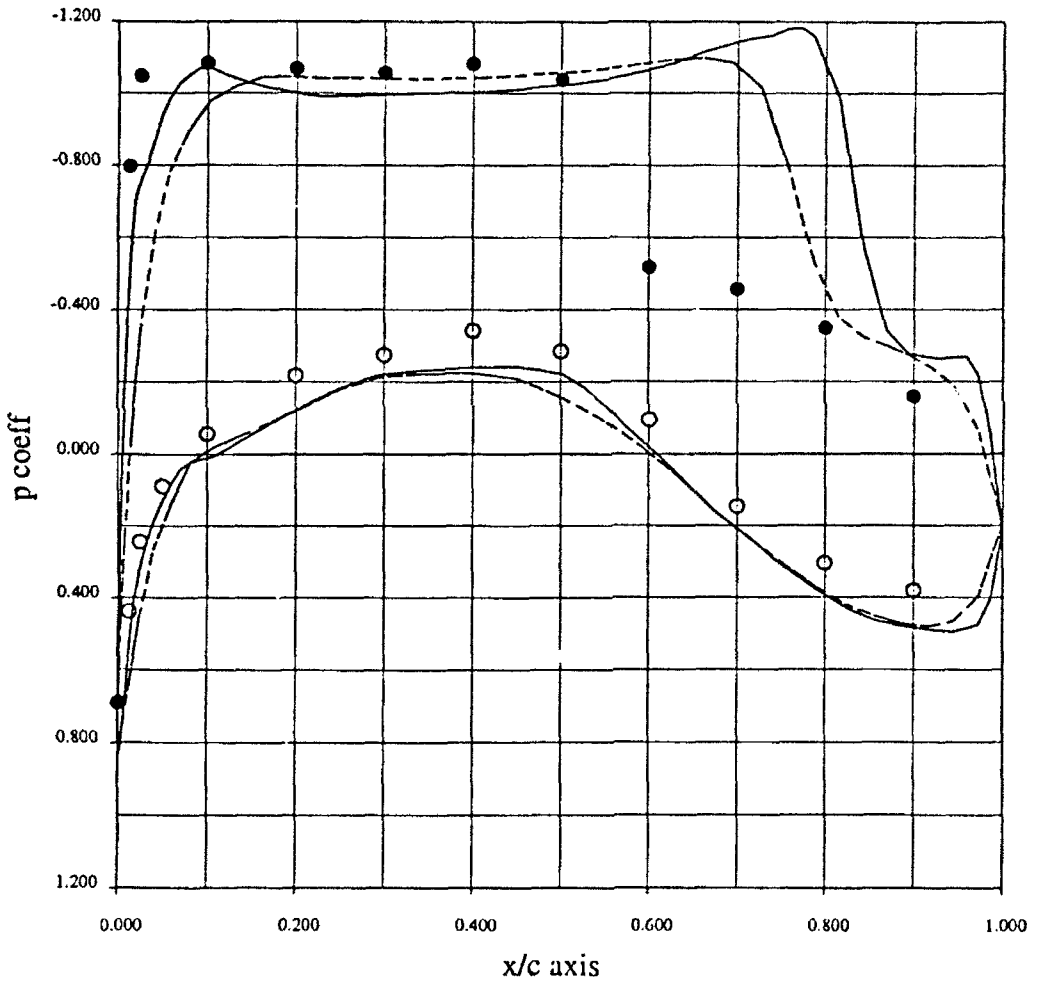







-1.200
-0.800
-0.400
0.000
0.400
0.800
1.200
p coeff
0.000
0.200
0.400
0.600
0.800
1.000
x/c axis


-1.200
-0.800
-0.400
0.000
0.400
0.800
1.200
p coeff
0.000
0.200
0.400
0.600
0.800
1.000
x/c axis


-1.200
-0.800
-0.400
0.000
0.400
0.800
1.200
p coeff
0.000
0.200
0.400
0.600
0.800
1.000
x/c axis